2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Sim-to-Real Transfer of Bolting Tasks with Tight Tolerance

Dongwon Son, Hyunsoo Yang and Dongjun Lee

Abstract— **In this paper, we propose a novel sim-to-real framework to solve bolting tasks with tight tolerance and complex contact geometry which are hard to be modeled. The sim-to-real has desirable features in terms of cost and safety, however, that of the assembly task is rare due to the lack of simulator, which can robustly render multi-contact assembly. We implement the sim-to-real transfer of nut tightening policy which is adaptive to uncertain bolt positions. This can be realized through developing a novel contact model, which is fast and robust to complex assembly geometry, and novel hierarchical controller with reinforcement learning (RL), which can perform the tasks with a narrow and complicated path. The fast and robust contact model is achieved by utilizing configuration space abstraction and passive midpoint integrator (PMI), which render the simulator robust even in a high stiffness contact condition. And we use sampling-based motion planning to construct a path library and design linear quadratic tracking controller as a low-level controller to be compliant and avoid local optima. Additionally, we use the RL agent as a high-level controller to make it possible to adapt to the bolt position uncertainty, thereby realizing sim-to-real. Experiments are performed to verify our proposed sim-to-real framework.**

I. Introduction

In general, reinforcement learning (RL) requires lots of data. Therefore, using simulation is safer and more cost-efficient than using many of expensive hardware. Also, if RL is in the simulation environment, we can access various sensing data which is used for result analysis or evaluating the effectiveness of each sensor. We can generate complex behavior and control strategy even for new mechanical designs under development, which is hard to generate real data.

Despite recent progress on the sim-to-real algorithm with domain randomization [1]–[3] and successful implementation including quadruped [4], [5], non-prehensile manipulation [1], [6], in hand manipulation [3], walking [7], results on assembly with complex motion and tight tolerance are rare. The main bottleneck of applying the sim-to-real framework of such a tight assembly task is a lack of simulator which RL can be applied to [8]. To apply RL to those tasks, robust multi-contact simulation between strongly non-convex meshes is necessary, because unreliable multi-contact solution results in the penetration which can make infeasible trajectory samples and degenerate RL performance. If we consider the computational efficiency of the multi-contact model, there are fewer options. In particular, bullet [9] and MuJoCo [10] are the widely used simulator for RL, however, nut tightening is hard to be simulated in real-time with these simulators, because of limitation on the contact model. Collision detection (CD) of these simulators is the main bottleneck of the whole simulation process in the case of assembly between non-convex shapes. It is based on searching through the mesh, and this makes the algorithms not to be scalable to the number of faces in the mesh.

Research supported by the Industrial Strategic Technology Development Program (20001045) of the Ministry of Trade, Industry & Energy (MOTIE).

The authors are with the Department of Mechanical & Aerospace Engineering and IAMD, Seoul National University, Seoul, Republic of Korea. D. Son is currently with Samsung Research, Seoul, Republic of Korea. {son3933,yangssoo,djlee}@snu.ac.kr. Corresponding author: Dongjun Lee.

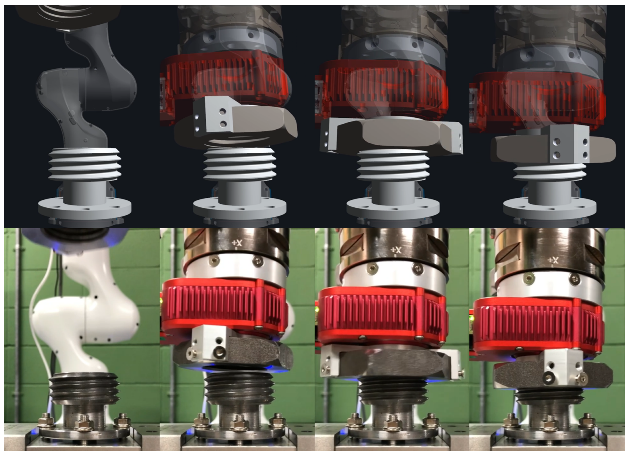

Fig. 1: Snapshots of sim-to-real transfer for bolting task

In this paper, we propose a novel sim-to-real transfer framework for bolting tasks, which has tight bolt-nut geometric tolerance (e.g., 0.3mm) and requires a complicated strategy to adapt geometric uncertainty in practice. More precisely, we propose a novel contact model, which, utilizing the configuration space (C-space) formulation, can substantially improve the robustness of contact enforcement (e.g., mitigated penetration) while providing the CD efficiency of the C-space. Furthermore, we devise a hierarchical controller, which consist of nominal low-level linear quadratic tracking (LQT) and high-level RL to adapt the geometric variability, rendering the control strategy robust against the issue of local optima [11] as well as the bolt-nut positioning uncertainty in practice (see Fig. 2). The proposed sim-to-real framework has also been experimentally verified with the real Franka Emika PANDA robot for the M48 bolt-nut assembly task. To our knowledge, our proposed framework is the very first sim-to-real result for bolting tasks with tight tolerance.

The main idea of the proposed contact model is utilizing C-space formulation and it has been studied in various areas. In [12], CD is accelerated by constructing C-space collision boundary with a data-driven model and labels from standard CD library, to be used in motion planning. On the other hand, our study not only focuses on CD but also extends C-space formulation to the calculation of the interaction force. In [13], they also calculate interaction force with C-space formulation to render realistic joint limits of the human, however, this approach cannot render robust multi-contact simulation in the case of tight assembly with complicated geometry because of

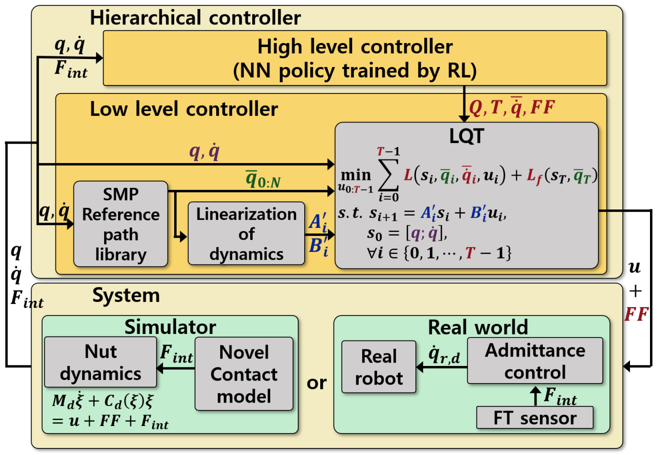

Fig. 2: Structure of the proposed hierarchical controller. Hierarchical controller receives state q, velocity $\dot{q}$ in C-space and interaction force F_{int} and calculates input u and feed-forward force FF to system. High-level controller decide parameter of LQT $(Q, T, \bar{q})$ and FF, where Q is scale of quadratic metric in the cost of L, L_f. The low-level controller calculates emulated input u in W-space, generating reference trajectory $\bar{q}$ from the SMP path library. The proposed hierarchical controller is trained in the simulator and transferred to the real robot as a sim-to-real manner.

numerical instability induced by discretization. We develop a novel volumetric state representation method in C-space which substantially improves robustness. Additionally, we introduce friction in C-space formulation, which is absent in previous works despite its importance in assembly.

The rest of the paper is organized as follow. Sec. II explains system modeling and background. In Sec. III, a novel multi-contact simulation framework is proposed. In Sec. IV, the idea of hierarchical controller is presented and Sec. V explains the experimental results and details on implementation in simulation and real robots. Finally, Sec. VI highlights our conclusion and future works.

II. Preliminary

A. System Description

In this paper, we utilize Franka Emika Panda, a 7-DoF robot arm. A specification of the bolt is M48 following KS-B-0201 [14]. We use an additional actuator, X-series of HEBI robotics, to rotate nut infinitely regardless of the joint limit of the last joint. A nut is rigidly attached to the additional actuator with a manufactured mounting part. The robot arm provides measurements of joint position and velocity. Force and torque values are measured by force-torque (F/T) sensor attached between end-effector and additional actuator. Regarding measurement, the objective task requires delicacy and we found that there is sensing uncertainty, which is turned out to induce jamming of the nominal controller without incorporating this uncertainty tolerance (see Sec. IV-A). Uncertainty issue can be resolved by RL (see Sec. IV-B).

Dynamics of the robot arm with n_r degree-of-freedom (dof) can be represented by Euler-Lagrange dynamics as follow

$$M(q_r)\ddot{q}_r + C(q_r, \dot{q}_r)\dot{q}_r + g(q_r) = \tau_r + \sum_i J_i^T F_{int,i}$$

where $M(q_r) \in \Re^{n_r \times n_r}$ is the inertia matrix, $C(q_r, \dot{q}_r) \in \Re^{n_r \times n_r}$ is Coriolis matrix, $g(q_r) \in \Re^{n_r}$ is gravity vector, $q_r \in \Re^{n_r}$ is the joint angle of the robot arm, $\tau_r \in \Re^{n_r}$ is joint torque, $F_{int,i} \in \Re^3$ is contact forces expressed in Cartesian contact frame, $J_i \in \Re^{3\times n_r}$ is jacobian matrix from robot joint space to Cartesian contact space. Note that the contact force at a contact point only includes force components without torque.

Since assembly tasks can be expressed more directly in the workspace (W-space), we shape robot dynamics to nut dynamics in W-space with an adequate controller. In this paper, we utilize the admittance control with the F/T sensor on the robot arm. We choose this admittance control over the impedance control, since Franka Emika robot arm has non-negligible friction and backlash for bolting tasks with tight tolerance, while the robot motion rather slow compared to sensing and actuation delays. The frequency of admittance control is 1kHz. We assume that the nut dynamics in SE(3) is shaped from the dynamics of the robot arm to be

$$M_d\dot{\xi} + C_d(\xi)\xi = u + \sum_i {J_{d,i}}^T F_{int,i} \tag{1}$$

where $M_d \in \Re^{6\times 6}$ is desired inertia matrix, $C_d(\xi) \in \Re^{6\times 6}$ is desired Coriolis matrix, $\xi = (v; \omega) \in se(3)$ is twist with velocity $v \in \Re^3$ and angular velocity $\omega \in \Re^3$, $u \in \Re^6$ is emulated desired input and $J_{d,i} \in \Re^{3\times 6}$ is jacobian matrix to nut frame. We use SE(3) space of (1) as W-space.

For the nut C-space representation, we adopt $q = (p; \phi) \in \Re^6$ where $p \in \Re^3$ is center position of the nut and $\phi \in \Re^3$ is Euler angle parameterization. Thus, conversion from the twist in the W-space to the velocity in the C-space is achieved by jacobian matrix with the relation of $\dot{q} = J_{cw}\xi$, where $J_{cw} \in \Re^{6\times 6}$.

B. Multi-Contact Simulation

Typically, contact solver is formulated in W-space and it requires contact points and geometric information of objects. Yet, for bolting, contact geometry is complex because the shape of the bolt and nut is complicated, thus the elaborate mesh which has a large number of faces should be needed to describe its geometry. Then mesh-mesh CD should be applied, but it is rather slow with the large size of the mesh. Furthermore too many contact points prone to generate unreliable normal vectors, and if contact clustering is applied to reduce the number of the contact point, it possibly ill-conditioned and physically inaccurate [8]. To avoid this, we will formulate the contact in the C-space with $q \in \Re^6$ as defined in Sec. II-A, then, the object is represented by a point with constraints boundary in this C-space, thus, the CD is achieved efficiently by evaluating $h(q)$ value (i.e., contact-off if interior, contact-on if exterior). Yet, friction is not straightforward to formulate in this C-space. Thus, we convert this contact information in C-space to contact points in W-space and apply contact solver which has been studied well [10], [15], [16]. We further integrate this novel contact model into passive midpoint integrator (PMI) [17], which

Algorithm 1: Training algorithm for implicit function

```
Result: $h(q)$
1  $S = \emptyset$
2  Get set of path nodes $S_{SMP}$ from SMP
3  while not converged do
4      $S = S \cup$ {set of uniformly distributed samples}
5      $S = S \cup$ {set of normally distributed samples
        from $S_{SMP}$}
6      $S =$ RejectionSampling($S$)
7      $S_{iv}, S_v =$ Evaluation($S$)
8      $S_{iv}, S_v =$ Balancing($S_{iv}, S_v$)
9      $h(q) =$ Training($S_{iv}, S_v$)
10 end
```

can allow us to simulate stiff and light objects with short computation time. Thus PMI not only allows us to collect a large amount of data within short wall time (see Sec. V-A) but also ensure the stability of the high-stiffness multi-contact simulations for the bolting operations.

III. Multi-Contact Simulation in C-space

A. Pfaffian Constraint Learning

To explain the proposed contact model, we first construct the surface of the contact volume in the 6D C-space. More precisely, we build a smooth function $h(q) \in \Re$ which is positive for contact-off state (i.e., interior of volume) or negative for contact-on state (i.e., exterior of volume). In other words, for all interior and exterior configuration states q_{in} and q_{ex}, $h(q)$ should satisfy $h(q_{in}) > 0$, $h(q_{ex}) < 0$ respectively. Then, $h(q) = 0$ functions as a boundary surface that determines collision of states in the C-space. Once we construct this $h(q)$, its gradient $\partial h/\partial q$ can be calculated, which encodes the normal direction of this surface and also is often used as the vector of Pfaffian constraint $A(q)$ for constrained dynamics to compute constraint force. To construct this $h(q)$, in particular, we apply a neural network (NN) approximation, which ensures the smoothness of $h(q)$.

NN is trained with labeled data from standard CD library such as bullet [9] and FCL [18]. To obtain labeled data set, the simplest approach is to generate uniform samples in bounded state space, and evaluate each point using the CD library. However, with this uniform distribution alone, it is difficult to learn the narrow and complex shape $h(q)$, which is important for bolting simulation. To construct accurate $h(q)$, samples that are densely distributed in adequate area play an important role, and it is difficult to sufficiently express this distribution with a uniform distribution. To solve this problem, we utilize the sampling-based motion planning (SMP) and the rejection sampling method. SMP generates trees of the path from a random initial state to the goal state. All these paths include the states that the nut undergoes during performing the assembly task, thus the samples along this path help to improve the preciseness of $h(q)$.

The procedure is summarized in algorithm 1 and details are as follows. First, we generate reference path nodes from goal state to bounded space in C-space through SMP algorithm (line 2). After that, we generate uniformly distributed samples from the bounded range (line 4), and add samples which are located near the generated SMP nodes (line 5). This can be realized by generating samples from normal distributions whose mean is each SMP node state. From this additional data set, we can get dense samples near the narrow assembled area. To improve the precision of the NN, we introduce a rejection sampling technique [19], that removes the sample if the value evaluated by the previously trained model $h(q)$ is far from zero (line 6). Note that the value evaluated by $h(q)$ close to 0 means that the sample is close to the boundary between interior and exterior. In this way, we can filter the samples, leaving only data that helps to improve accuracy of $h(q)$.

For training, we assign a value of 0.5 for the interior data and -0.5 for the exterior data (line 7). Labeled data is balanced from removing randomly chosen samples in a class that has more samples than another (line 8). Then, NN is trained with L1 loss (line 9). We repeat these procedures until convergence.

Thanks to the trained function $h(q)$, we can efficiently determine the contact condition with states expressed in C-space, and this idea is similar to the work in [12], which uses constructed $h(q)$ as CD for motion planning. However, we use the implicit function for not only CD but also calculating Pfaffian constraint direction A^T and interaction force. The details on how we calculate interaction force are explained in the following section.

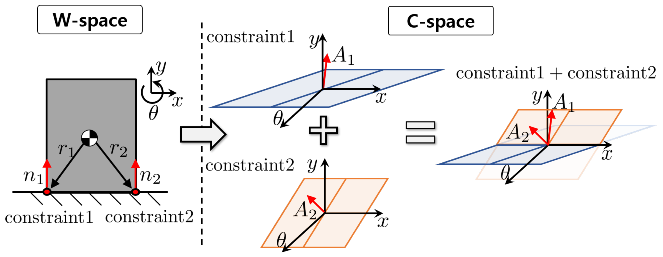

Fig. 3: Illustration of multi-Pfaffian constraints in 2D box example. Left shows constraints in W-space and right shows converted constraints in C-space and implicit surface by combining two constraints.

B. Robust Contact Model in C-space

To understand the main idea of this section, let us explain with an example of a 2D box on the floor as described in Fig. 3. First, we explain how the contact constraints in W-space can be converted to Pfaffian constraints in C-space, and how the ideal implicit surface looks like in C-space. To express the condition of contact without penetration in this example, we need at least 2 contact points as illustrated in the left of Fig. 3. If the constraint consists of one contact point, we always find the direction of motion which can make penetration. These two contact constraints in W-space can be converted to normal vector in C-space, $A_i^T := (n_i; A_i^\tau) \in \Re^3$, where $n_i \in \Re^2$ is normal vector of i-th contact points and $A_i^\tau \in \Re$ is converted moment of i-th constraint. Each constraint has corresponding plane as illustrated in the right of Fig. 3. The combination of two surfaces consists of the ideal implicit surface $h(q) = 0$.

Returning to our problem, the nut appears as a point in C-space and surface $h(q) = 0$ can have a complicated

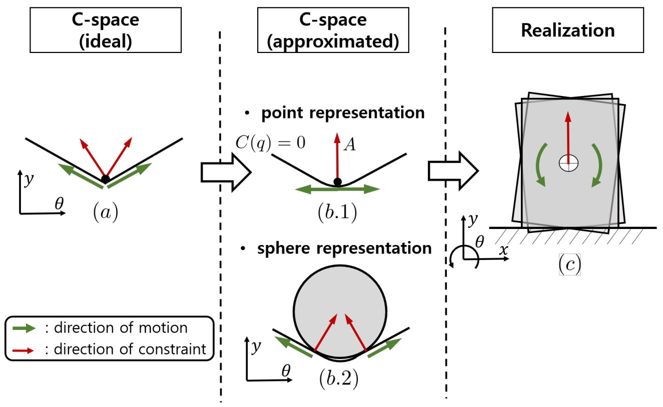

Fig. 4: Illustration of relation between motion, constraints and implicit surface in ideal C-space (left), in approximated constraint with each representation method (middle) in 2d box case. Constraints and motion in C-space can be realized as box motion with the floor constraint (right).

shape, which is a combination of constraint surfaces, since each contact surfaces is constructed from complex contact points in W-space. With $h(q)$ constructed in Sec. III-A and friction-less condition, contact force can be computed with following procedures. With Pfaffian constraint matrix in C-space A which is gradient of $h(q)$, discretized dynamics of (1) satisfies following equation in the notion of PMI [17],

$$\xi^+ = \hat{M}^{-1}(M_d - \frac{1}{2}C(w^-))\xi^- + \Delta t\hat{M}^{-1}u + \Delta t\hat{M}^{-1}J_{cw}^TA^T\lambda \quad (2)$$

where $\xi^+ \in \Re^6$ is twist of the next time step, $\xi^- \in \Re^6$ is twist of the current time step which consists of linear velocity $v^- \in \Re^3$ and angular velocity $w^- \in \Re^3$, $C(w^-) \in \Re^{6\times6}$ is Coriolis matrix, Δt is the time interval, $\hat{M} = M_d + \frac{1}{2}C(w^-)$, λ is Lagrange multiplier of constraint force. Then, we can obtain projected dynamics along the constraint direction A^T as

$$v_A^+ = \bar{b} + \bar{A}\lambda \quad (3)$$

where $v_A^+ = AJ_{cw}\xi^+ \in \Re$ represents velocity of next time step along A^T, $\bar{A} = \Delta tAJ_{cw}\hat{M}^{-1}J_{cw}^TA^T \in \Re$ and $\bar{b} = AJ_{cw}\hat{M}^{-1}(M_d - \frac{1}{2}C(w^-))\xi^- + \Delta tAJ_{cw}\hat{M}^{-1}u \in \Re$. As mentioned above, we assume zero friction in this section, therefore interaction force F_{int} can be obtained from the complementary condition [20] $0 \le v_A^+ \perp \lambda \ge 0$ and relation $F_{int} = A^T\lambda$.

Note that, ideally, it is possible to have multiple constraints with one point state on the implicit surface in C-space as (a) of Fig. 4. However, in our case, the implicit function is approximated to smooth NN as (b.1) in Fig. 4, so that the interaction force can be calculated with only one constraint in the case of a point representation as derived above. In the study of [13], this formulation is used for calculating the interaction force. However, multi-contact simulation cannot be properly rendered with a point representation in terms of robustness.

The problem is that the discretization is necessary for this time stepping formulation with Δt, and the state of the next time step with the constraint of the current state can enter into the exterior area (i.e., $h(q) < 0$) after Δt. It results in constraint violation and penetration (see (b.1) and (c) in Fig. 4). Penetration is critical to tight bolting tasks because it can result in an infeasible trajectory. For example, if there is severe penetration in nut tightening simulation, the nut can pass through the thread of bolt without rotation. Also, the penetration provides inaccurate interaction force which is one of the important inputs that determines RL performance. Thus we should resolve the penetration issue to apply the sim-to-real framework.

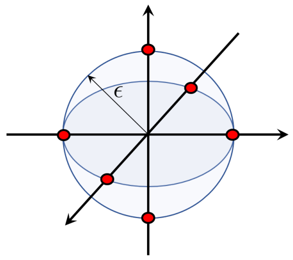

Fig. 5: Sphere approximation in 3D case.

To solve this issue, we formulate multi-Pfaffian constraints by introducing a volumetric sphere approximation. In other words, a state is represented by a center of a sphere in C-space and multiple contacts are represented by contacts between the sphere and the surfaces. The sphere is defined as

$$\mathcal{S}_q = \{\, q' \in \Re^6 \mid \|q' - q\| = \epsilon,\ q \in \Re^6\},$$

where q is the center of the sphere which is given state in C-space. If only one point at q in C-space is used in simulation, it can chatter and violate constraints ((c) in Fig 4). Yet, with the sphere model, it has more contact points at the same time, among which some of these contact points are possible to be redundant with each other (i.e., representing the same constraint). Thus, if one is temporarily violated during simulation, other points will still active, thus, overall, contact can be more stably enforced. In other words, by increasing the redundancy of contact points in C-space, we enhance the robustness of the contact enforcement. The comparison experiment between point and sphere representation is presented in Sec. V-B. In practice, the sphere can be approximated to a set of points on each axis. That is, the sphere in C-space can be approximated by 12 points since C-space is 6D and each dimension axis represented by 2 points (see Fig 5 for approximation in 3D case).

Even with this robustly enforced contacts, the issue of friction generation in the C-space remains. This issue is resolved by converting the contact points in the C-space to contact points in W-space, and using standard constraint-based contact solver [10], [15], [16], which generates contact force with friction. This process is discussed in Sec. III-C.

C. Space Conversion to W-space and Friction Model

To generate friction, we first convert C-space Pfaffian constraints A_i to W-space constraints $\hat{A}_i$, and find contact points r_i and normal n_i, which can produce $\hat{A}_i$. To convert the contact normal A_i in the C-space (between the C-space

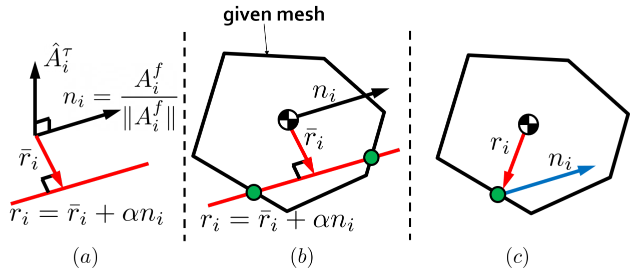
given mesh
$\hat{A}_i^\tau$
$n_i = \frac{A_i^f}{\|A_i^f\|}$
$\bar{r}_i$
$r_i = \bar{r}_i + \alpha n_i$
n_i
$\bar{r}_i$
$r_i = \bar{r}_i + \alpha n_i$
r_i
n_i
(a)
(b)
(c)

Fig. 6: Illustration of conversion from the constraint of W-space A_i to the contact point r_i, n_i. In (a) relation between components of constraint in W-space and those in contact space is described. In (b) method of finding candidate points is described. In (c) identified r_i and n_i are described.

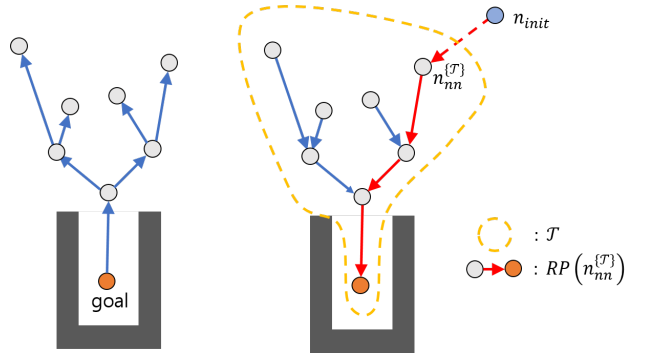
n_{init}
$n_{nn}^{\{\mathcal{T}\}}$
goal
: $\mathcal{T}$
: $RP\left(n_{nn}^{\{\mathcal{T}\}}\right)$

Fig. 7: Illustration of making SMP tree (left) and finding reference path (right).

contact sphere and the contact surface $h(q) = 0$) into the contact points on the W-space nut object, we notice the fact that, for friction-less contact, the Cartesian contact force direction for the bolt object with the i-th contact point is given by $n_i = A_i^f/\|A_i^f\|$ where $(A_i^f, A_i^\tau) := (\partial h/\partial q)^T$, with $A_i^f, A_i^\tau \in \Re^3$. This Cartesian contact force should be exerted to the point on the nut object. Furthermore, since the W-space contact is based on the point contact model, this point on the nut object should be able to produce not only this Cartesian contact force, but also the contact moment, whose direction is given by $J_{cw}^{w\ T} A_i^\tau := \hat{A}_i^\tau$, where $J_{cw}^w \in \Re^{3\times3}$ is bottom right corner matrix of J_{cw}.

Now, the remaining process to convert A_i to contact points is to find such a point on a mesh of the bolt object, that can produce both the contact normal n_i and the contact moment $\hat{A}_i^\tau$. We can decompose r_i to $\bar{r}_i + \alpha n_i$, where $\bar{r}_i$ is a component perpendicular to n_i with $\alpha \in \Re$ as illustrated in Fig. 6. And then, with the relation of $\hat{A}_i^\tau = r_i \times n_i$, we can find general solution of r_i as follow,

$$r_i = \frac{n_i \times \hat{A}_i^\tau}{\|n_i\|^2} + \alpha n_i. \tag{4}$$

Next, we should identify α of each constraint to decide contact points. Identifying α means finding contact points, which are on both the object mesh and the line vector defined from (4). This process is illustrated through (b) to (c) in Fig. 6. First, we can find faces on the object mesh with which the line intersects, and calculate candidate points as Fig. 6 (b). After finding candidates, we should filter them out using adequate conditions. In particular, we already know the normal direction n_i and it should point toward inside of the mesh. So among candidates, we can rule out that with the sign of inner product between the normal vector of faces and n_i as shown in the green point in Fig. 6 (c). We then use these W-space contact points into the contact solver (e.g. [10], [15], [16]). In particular, we use [15] in the experiments.

We found that some of these W-space contact points are located outside the bolt object. We inject still these points into the contact solver as well, since, even though they are not on the object, their usage is found to substantially improve the contact robustness and mitigate the penetration. We conjecture that this outside point may play a role similar to "instantaneous center of rotation", which is virtual and not real, yet, can be compatibly incorporated (or exploited) into the physics of the real objects. Further investigation on the interpretation and implementation ramifications of utilizing these "outside" contact points is a topic for future research.

IV. Hierarchical Controller

After constructing the fast and robust multi-contact simulator, we need to design a proper controller for tight assembly tasks. These tasks are hard to be automated because path to the goal is narrow and there is sensing uncertainty in real robot, so sticking and jamming can be raised often. We solve these issues using a hierarchical control: 1) SMP based low-level controller, and 2) high-level controller trained by RL. The low-level controller contributes to being compliant and avoiding local optima, and the high-level controller makes it more robust to the bolt position and orientation uncertainty. In this section, two controllers are explained sequentially.

A. Low-Level Controller

To understand the low-level controller proposed in this paper, a path library strategy from SMP needs to be understood first. The key idea is that, in most of the assembly tasks, the goal position is fixed and the success of the task is determined by whether the goal position is reached. Thus, with SMP whose root is the goal, we can obtain the various path toward the open space and this tree structure can be used as a path library. Furthermore, the probabilistic completeness property of SMP is suited to the bolting task since a path to the goal is narrow and complex in C-space. Specifically, we spread tree from goal to bounded C-space, then we can get a path library that can connect any nodes $n^{\{\mathcal{T}\}}$ in the SMP tree $\mathcal{T}$ to goal. Let us define the realized reference path starting from $n^{\{\mathcal{T}\}}$ as $RP(n^{\{\mathcal{T}\}})$. However, in most cases, the initial state does not coincide with one of the nodes in the SMP tree $\mathcal{T}$. To address this generalization problem, we find the nearest node $n_{nn}^{\{\mathcal{T}\}}$ from given initial position n_{init} and add additional nodes to $RP(n_{nn}^{\{\mathcal{T}\}})$ to connect between n_{init} and $n_{nn}^{\{\mathcal{T}\}}$. These processes are illustrated in Fig. 7. Note that SMP tree $\mathcal{T}$ used in this section can be shared with the one used in Pfaffian constraint learning in Sec. III-A. After finding a reference path, we can design a path tracking controller.

We design a tracking controller using discrete time finite

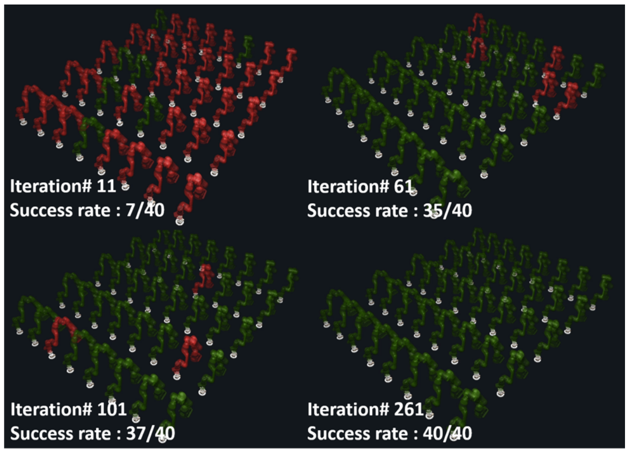

Fig. 8: Visualization of success rate change with increasing iteration number. Green robots represent success of the task and red robots represent failure.

horizon LQT. This control problem can be formulated as

$$\begin{aligned} &\min_{u_{0:T-1}} \sum_{i=0}^{T-1} L(s_i, \bar{q}_i, \bar{\dot{q}}_i, u_i) + L_f(s_T, \bar{q}_T) \\ &\text{subject to. } s_{i+1} = A'_i s_i + B'_i u_i, \quad s_0 = s_{init}, \\ &\qquad \forall i \in \{0, 1, \cdots, T-1\} \end{aligned} \tag{5}$$

where $s = [q; \dot{q}] \in \Re^{12}$, $q \in \Re^6$ is state in C-space, $\dot{q} \in \Re^6$ is velocity in C-space, $q_{init} \in \Re^6$ is a given initial state, $\bar{q} \in \Re^6$ is reference state from the path library, $\bar{\dot{q}} \in \Re^6$ is reference velocity from high-level controller, $u \in \Re^6$ is emulated desired input in (1), T is length of horizon, $A' \in \Re^{12\times12}$, $B' \in \Re^{12\times6}$ are system matrix from linearization of admittance dynamics along reference state, L is intermediate cost, L_f is final cost. Note that admittance dynamics is projected from W-space to C-space, then linearized. The algorithm for solving LQT problem is explained in [21] and we can get a solution of input trajectory $u_{0:T-1}$ from the solver.

So far we have discussed how to obtain a reference path and the path tracking controller. These procedures are fast enough to respond to external perturbation in real-time. It means that if the state is perturbed by an external interaction, the low-level controller can replan a reference path starting from a perturbed state and calculate input to follow it in every time step. This is thanks to efficient algorithms of finding a reference path using SMP tree and computationally efficient formulation of LQT.

Although the low-level controller can respond to the disturbances in real-time, it cannot address the uncertainty of the environmental geometry. For the case of nut tightening, if the bolt position is shifted to 3mm, then tightening fails because the low-level controller incorporates only kinematics while the impedance of the controller is fixed. This limitation of the low-level controller motivates to design a high-level controller.

B. High-Level Controller

We can solve geometric uncertainty problem by introducing the high-level controller. The idea is that if we can find the proper compliance at every time step considering progress of the task and sensing feedback, the controller can adapt an uncertain bolt position. However, this high-level controller is hard to be achieved by the analytical method. Thus, we use RL with domain randomization [1].

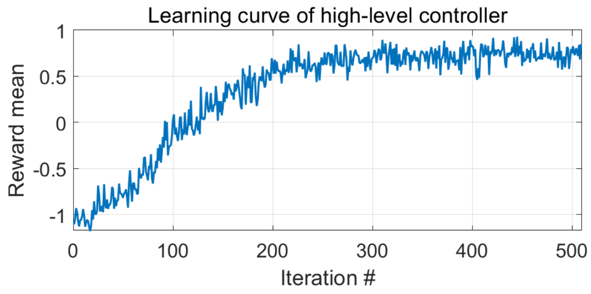

Fig. 9: Learning curve of training high-level controller

The outputs of the RL agent are parameters for LQT controller and feedforward force given inputs as the state, interaction force, and reference path. The feedforward force is necessary for adopting behavior like pushing the nut to the bolt. It is notable that we can set sparse costs such as +1 only in case of success, because the low-level controller is already well-functioning. In our experiment, the RL agent successes a few trials even in the first iteration of training.

It is also worth mentioning that, although the proposed contact model is obtained by NN approximation, it is only the abstraction of contact geometry, thus we can easily modify simulation parameters including friction, mass, and offset of environment. On the other hand, with the methods which approximate whole transition model such as [22], [23], NN should be trained again when modifying dynamics properties. This enables us to apply domain randomization eaisly. In our experiment, a naive uniformly distributed randomization of bolt position and orientation is enough to address sensing uncertainty during the sim-to-real transfer.

V. Experiments

A. Implementation Details

In this section, the implementation details of the proposed contact model and the hierarchical controller are explained sequentially. Regarding Paffian constraint learning in Sec. III-A, we use FCL [18] as ground truth of CD between the nut and bolt mesh because it provides mesh-mesh CD. Note that FCL is used only for learning $h(q)$, not in the nut tightening simulation. It means that $h(q)$ can be regarded as an abstraction of FCL result in W-space into C-space.

To implement the low-level controller, we need an SMP tree, and RRT* implemented in OMPL [24] is used to generate it. SMP is carried out in C-space. 46,000 nodes are generated and saved as a tree structure whose root is the goal of the task. Finding a reference path to the goal is trivial thanks to this directed tree structure. After finding the reference path, admittance dynamics (1) is converted to C-space, and linearized and discretized along with reference nodes, then we can get the dynamics form in (5), and input u is calculated from LQT formulation. Total execution time including searching the nearest node, finding the path to root, and solving the LQT problem is about $0.2ms$.

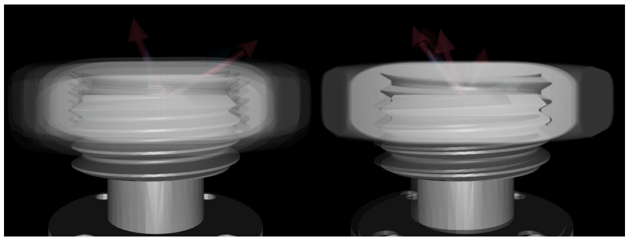
Fig. 10: Result of comparison experiment between a point representation (left) and sphere representation (right) to evaluate performance of robustness.

method	FCL (CD)	proposed (CD)	FCL (CD+normal)	proposed (CD+normal)
average	2.81	0.006	12.1	0.132
max	21.3	0.017	21.6	0.328
min	0.016	0.004	4.05	0.102

TABLE I: Calculation time taken to conduct CD and to obtain normal vectors in milliseconds for each method.

Next, the implementation of RL policy as a high-level controller is explained. The inputs of the high-level controller are current configuration q, velocity $\dot{q}$, a stack of 3 nearest states in reference path, interaction force F_{int}, and vector from the current state to the goal (total 42 dimensions). Actions are a scale of quadratic metric for an error of translational state from reference path ($Q_{pos,tran}$), that of Euler angle error ($Q_{pos,euler}$), that of linear velocity ($Q_{vel,linear}$), that of angular velocity ($Q_{vel,angular}$), LQT time horizon (T), number of reference node skipping in reference path, LQT reference tracking velocity, feedforward force and torque (FF) (total 13 dimensions). To train the high-level controller, we use PPO [25] which is a state-of-the-art on-policy algorithm suitable with our efficient simulator. We also apply the domain randomization technique to adapt geometric uncertainty. The bolt position is changed with uniform distribution with a fixed range, and the position is maintained in each episode. We train policy during 270 iterations with about 3.2×10^6 time-steps as Fig. 8 and 9. It takes about 3 hours in wall time which is comparably short thanks to the proposed fast simulation and parallel rollout.

B. Experiments of Contact Model

We conduct a comparison experiment to evaluate the robustness of each contact representation (point and sphere in III-B). The evaluation is conducted by visualizing the simulation environment in which random disturbances are applied with the nut in each representation, and then comparing how severe the penetration is. The result of visualization is that, as seen in Fig. 10, the nut severely penetrates the bolt in the point representation (left in Fig. 10), however, the penetration is hardly shown in the sphere representation (right in Fig. 10). This result shows that introducing sphere representation improves the robustness of multi-contact simulation.

Additionally, we compare the efficiency of CD with popular off-the-selves CD library to evaluate the calculation efficiency of the proposed contact model. We compare the CD performance of the proposed algorithm with FCL, which can be representative CD library, because FCL uses libccd [26] in case of CD between non-convex mesh, and libccd is also widely used in other libraries such as MuJoCo and Bullet. The evaluation was carried out by calculating the average, maximum, minimum time during the 13,000 step calculation at the condition that the bolt and nut are fastened and perturbed. The number of faces in the bolt and nut mesh is 5,020 and 5,972 respectively. 4 methods are compared: 1) CD using FCL, 2) CD and calculation of contact information such as penetration depth and normal vectors using FCL, 3) CD using the proposed method, 4) CD and calculation of Pfaffian constraints A using the proposed method. Comparison between 2) and 4) is meaningful in terms of that both calculated information is necessary for the contact solver. The computation environment is AMD® Ryzen 7 3700x of which the base clock is 3.6GHz with 8 cores.

The results are shown in table I. CD through our algorithm is about 460 times faster than the CD of FCL, and if including calculation of contact information, ours is 90 times faster than that of FCL. This result is because CD is done by one calculation of $h(q)$ on the proposed model and the calculation is independent of the size of the mesh, while FCL is based on searching through the mesh, then it gets slow down as larger the size of the mesh.

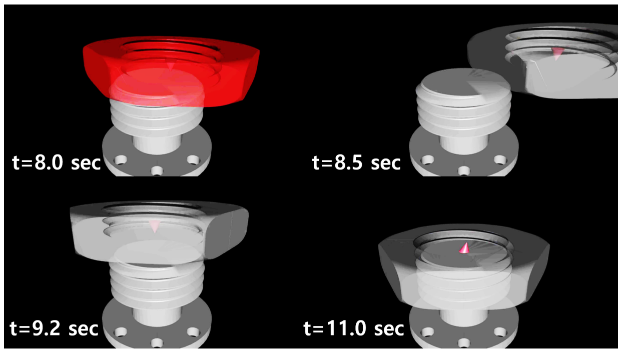

Fig. 11: Snapshots of showing reactiveness of low-level controller. The controller enable the nut to recover from a disturbed state and resume bolting operations.

C. Experiments of Hierarchical Controller

We qualitatively evaluate the reactiveness of the low-level controller by adding disturbances as it performs its task. As seen in Fig. 11, the controller can replan the path with a perturbed state in real-time and follow back to the replanned reference path. Reactiveness is also verified in the experiment of the real robot. This test also can be seen in the attached video.

We also conduct an experiment to evaluate the limitations of bolt position uncertainty in which the controller can succeed, first in a simulation environment as a sim-to-sim manner. We compare two controllers, low-level controller and hierarchical controller. Offsets from 0.0mm to 4.0mm with 1.0mm interval are applied to bolt position, and both controllers carry out the task 10 times, and the success of the task is evaluated. The results are listed in table II. Proposed low-level controller successes tasks until 2.0mm shifting, but fails after 3.0mm shifting. On the other hand, the hierarchical controller succeeds tasks even with an offset larger than 3.0mm. This result verifies that the proposed hierarchical controller improves adaptiveness against the

Offset		Sim-to-sim		Sim-to-real	
d (mm)	θ (deg)	LL	HC	LL	HC
0.0	0	10/10	10/10	4/4	4/4
1.0	0	10/10	10/10	-	-
2.0	0	10/10	10/10	-	-
3.0	0	0/10	9/10	0/4	4/4
4.0	0	0/10	5/10	0/4	3/4
3.0	2	0/10	9/10	0/4	4/4

TABLE II: Counts of executions of low-level (LL) and hierarchical (HC) controller in each offset condition for simulation and real robot. d is linear offset along y-axis and θ is angular offset along x-axis.

geometric uncertainty of the environment. Next, we explain details about the sim-to-real.

Finally, the experiment in the real robot is conducted in a sim-to-real manner. We execute the hierarchical controller 4 times to the misaligned bolt and the results are listed in the table II. The hierarchical controller successes with all offset cases containing 4 mm translational offset and 2 degrees angular offset. These results show that the trained hierarchical controller is robust to uncertainty even in the real world, and this improvement of adaptiveness renders the sim-to-real transfer.

There are two major reasons for the improvement of the adaptiveness compared with the low-level controller. The first reason is incorporating the F/T sensor data. In the execution to shifted bolt, trained policy tends to slow down and explores the area near the entrance of insertion. In the simulation, if we remove contact force in the dynamics of the simulator, then the policy tends not to turn the nut. It means that F/T sensor data is important to decide the end of insertion and the start of turning. The second reason is the feedforward input in the high-level controller. The policy can guide nut to states which are not in the SMP tree due to this additional force term. Especially, a downward force is generated near the entrance of insertion which improves adaptiveness.

VI. Conclusion

We develop a novel sim-to-real framework to solve bolting tasks with tight tolerance and complex contact geometry with a robust and fast simulator and novel hierarchical controller structure. Robustness and calculation efficiency of the simulator is improved by utilizing C-space abstraction and applying the volumetric representation of the state. As a result, the developed simulator can run four times faster than in real-time with $1ms$ integration interval. Additionally, we develop a novel reactive SMP-based controller which can avoid local optima. Using this efficient simulation and reactive controller, we can efficiently train policy for bolting task with an on-policy RL algorithm. This policy is robust to the uncertainty of bolt position and orientation due to being trained with domain randomization. Furthermore, we verify our sim-to-real framework by zero-shot transfer to real robot and it is the very first result, to our knowledge.

The possible direction of future works includes: 1) rigorous analysis on contact model; 2) extension of the sim-to-real framework to multi-modal sensing; and 3) extension to general assembly tasks.

References

[1] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int'l Conference on Robotics & Automation*, 2018, pp. 3803–3810.

[2] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. IEEE Int'l Conference on Robotics & Automation*, 2019, pp. 8973–8979.

[3] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[4] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, pp. 1–13, 2019.

[5] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[6] R. Jeong, J. Kay, F. Romano, T. Lampe, T. Rothorl, A. Abdolmaleki, T. Erez, Y. Tassa, and F. Nori, "Modelling generalized forces with reinforcement learning for sim-to-real transfer," *arXiv preprint arXiv:1910.09471*, 2019.

[7] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion," in *Proc. IEEE/RSJ Int'l Conference on Intelligent Robots & Systems*, 2019, pp. 3503–3510.

[8] M. Kim, J. Yoon, D. Son, and D. J. Lee, "Data-driven contact clustering for robot simulation," in *Proc. IEEE Int'l Conference on Robotics & Automation*, 2019, pp. 8278–8284.

[9] E. Coumans, "Bullet physics engine," *Open Source Software: http://bulletphysics. org*, vol. 1, no. 3, p. 84, 2010.

[10] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. IEEE/RSJ Int'l Conference on Intelligent Robots & Systems*, 2012, pp. 5026–5033.

[11] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *Proc. of the IEEE Int'l Conference on Robotics & Automation*, 2018, pp. 3524–3531.

[12] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *IEEE Transactions on Robotics*, 2020.

[13] Y. Jiang and C. K. Liu, "Data-driven approach to simulating realistic human joint constraints," in *Proc. IEEE Int'l Conference on Robotics & Automation*, 2018, pp. 1098–1103.

[14] "Metric coarse screw threads," KS B 0201, Korean Standard, 2016.

[15] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[16] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, no. 3, pp. 231–247, 1997.

[17] M. Kim, Y. Lee, Y. Lee, and D. J. Lee, "Haptic rendering and interactive simulation using passive midpoint integration," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1341–1362, 2017.

[18] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Proc. IEEE Int'l Conference on Robotics & Automation*, 2012, pp. 3859–3866.

[19] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[20] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2016.

[21] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," in *Proc. American Control Conference*. IEEE, 2005, pp. 2275–2280.

[22] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *Proc. International Conference on Machine Learning*, 2018, pp. 4467–4476.

[23] A. Ajay, M. Bauza, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling, "Combining physical simulators and object-based networks for control," in *Proc. IEEE Int'l Conference on Robotics & Automation*, 2019, pp. 3217–3223.

[24] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[26] D. Fišer, "libccd," https://github.com/danfis/libccd.