2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Data-Driven Contact Clustering for Robot Simulation

Myungsin Kim, Jaemin Yoon, Dongwon Son, and Dongjun Lee

***Abstract*— We propose a novel data-driven learning-based contact clustering (i.e., of contact points and contact normals) framework for rigid-body robot simulation, with its accuracy established/verified by real experimental data. We first construct an experimental robotic setup with force/torque (F/T) sensors to collect real contact motion/force data. We then design a multilayer perceptron (MLP) network for the contact clustering based on the full motion and force/torque information of the contacts. We also adopt the constraint-based optimization contact solver to facilitate the learning of our MLP network during the training. Our proposed data-driven/learning-based contact clustering framework is then verified against the experimental setup, compared with other techniques/simulators and shown to significantly (or meaningfully) enhance the accuracy of contact simulation as compared to them.**

I. Introduction

Contact simulation (among rigid bodies) is one of the key components in any robot simulators. This problem of contact simulation has a long history of research (e.g., [1], [2], [3]) and still remains as an active area of investigation (e.g., [4], [5], [6]). This contact simulation is becoming even more important with the recent advent and flourishing of the data-driven robotics research, particularly those utilizing deep-learning technique (e.g., [7], [8], [9], [10]). This is because those data-driven techniques typically necessitate huge amount of data covering wide spectrum of operations (e.g., unstable fall-down), and collection of that using real robots is, in many cases, simply infeasible due to the concerns on cost, time, and safety and also due to the lack of affordable/implementable sensors to measure the data of importance.

For robotic applications with extensive contacts (e.g., peg-in-hole, assembly - see Fig. 1), such exploitation of simulation has been rather limited, since the contact simulation of many simulators is often not so stable and/or its accuracy not well-established [4]. The issue of instability, we believe, could be to some extent suppressed by using implicit integration schemes (e.g.,[11], [12]) or our recently-proposed PMI (passive midpoint integration) framework [13], [14], which enforces discrete-time passivity of rigid-body simulation. On the other hand, the accuracy of contact simulation, to a large extent, is left unestablished with thorough comparisons against experimental data being mostly missed. In this paper, we focus on this issue of accuracy of contact simulation for robot simulation. This accuracy, of course, is crucial to ensure the results of those data-driven techniques be transferrable from simulation to real-world robotic systems [15], [16].

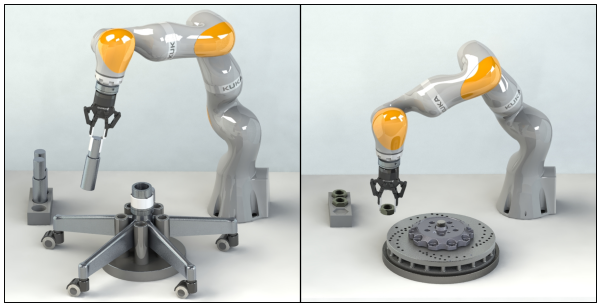

Fig. 1. Illustrative examples of robotic application where precise physical interaction is important: peg-in-hole and robot assembly.

The contact simulation typically consists of the following three procedures: (1) contact detection, (2) contact clustering (of contact points/normals), and (3) contact solving. The problem of contact detection has been more or less settled down with standard approaches available (e.g., [17], [18]). The problem of contact solving also boasts a large number of well-established techniques with the following two main categories: 1) constraint-based contact solving with linear complementarity problem (LCP) formulation (e.g., [1], [19], [3], [20], [21], [22]; also adopted by ODE [23]) or that with optimization formulation (e.g., [6], [24]; and [25], which renders formulation invertible with some regularization); and 2) penalty-based contact solving (e.g., [26], [27], [28]), which is typically simpler/faster than the constraint-based methods with the price of inter-object penetration.

The problem of contact clustering (i.e., clustering contact points and/or contact normals into a manageable number of clusters) is, however, much less investigated than the other two procedures. This contact clustering is imperative to achieve reasonably-fast speed of the simulation calculation while also enhancing numerical stability of the contact simulation (see Sec. II-B for more explanation). Some of rare, yet, to our knowledge, still representative works in this area are [26], [22], both standing upon the k-means clustering technique [29]. The work of [26] proposed a k-means contact clustering based on contact positions and penetration depths. This clustering is then used for the penalty-based contact force calculation to enhance interaction stability by reducing the number of contact points, which consequently limits the maximum contact stiffness. The work of [22] presented a k-means contact clustering in the seven-dimensional space of

Research supported by the Industrial Strategic Technology Development Program (20001045) of the Ministry of Trade, Industry & Energy (MOTIE) and the Global Frontier R&D Program on <Human-Centered Interaction For Coexistence> (NRF-2013M3A6A3079227) of the National Research Foundation (NRF) funded by the Ministry of Science and ICT (MSIT) of Korea.

Authors are with Department of Mechanical & Aerospace Engineering and IAMD, Seoul National University, Seoul, Republic of Korea. {myungsinkim,yjm5181,son3933,djlee}@snu.ac.kr

contact positions, contact normals, and friction coefficients, the result of which is then used for the LCP-based contact force calculation. They also reported that their seven-dimensional space clustering garners more stable contact than, e.g., the simple contact sorting of Gazebo [30]. As can be seen from these, the k-means clustering is rather the state-of-the-art for the contact clustering, which is also adopted by Bullet simulator [31] although its details not fully disclosed. Even so, none of these works explicitly establish the accuracy of their scheme against the real experimental data.

Along this line, in this paper, we propose a novel data-driven learning-based contact clustering framework for rigid-body simulation with its accuracy being established/verified by using real experimental data. Here, we believe the data-driven/learning-based approach is suitable, since the mapping from the contact clustering to the contact simulation accuracy is, in general, difficult (or, more often, impossible) to derive analytically. This is also motivated by our experience that, by hand-tuning the contact clustering, sometimes, the contact simulation results seem more plausible. For this, we first construct an experimental robotic setup with force/torque (F/T) sensors to collect real contact motion/force data. We then design a multilayer perceptron (MLP) network for the contact clustering, which utilizes the full motion (i.e., relative position and velocity) and force/torque information of the colliding rigid body. It is worthwhile to mention that this full motion/force/torque information is typically not used in the works of the contact clustering based on the k-means clustering (e.g., [26], [22]), although the contact behavior would likely depend on this full motion and force/torque data during the contact. For our framework, we also adopt the constraint-based optimization contact solver similar to [25], since, from its being invertible, it facilitates the learning of our MLP network during the training. Our proposed data-driven/learning-based contact clustering framework is then verified against the experimental setup along with other techniques/simulators and shown to significantly enhance the accuracy of contact simulation (e.g., see Table I).

The rest of the paper is organized as follows. We first introduce the rigid body contact model and the standard k-means contact clustering in Sec. II. We then present main result in in Sec. III, i.e., the data-driven/learning-based contact clustering framework. Contact simulation performance of our proposed framework is then compared against some open-source robot simulators and some standard k-means contact clustering techniques are provided in Sec. IV, following with some concluding remarks and comments on future research topics in Sec. V.

II. Preliminary

A. Rigid Body Contact for Robot Simulation

Contact is imperative for robotic application such as grasping, manipulation, and assembly etc. Thus, solving contact stably and accurately is crucial for the robot simulation. As mentioned in Sec. I, two methods are typically utilized to solve rigid body contact: constraint-based contact solving and penalty-based contact solving. The latter method is simply using a virtual spring between overlapping bodies to eliminate their inter-penetration. This is widely used due to its easiness in using and fast speed. However, it necessitates interpenetration to generate contact force and suffers from instability if it is applied for multiple contact points [26] and/or uses large spring gain without, e.g., implicit formulation [14], [11].

On the other hand, the constraint-based method determines appropriate contact force analytically. First of all, for instance, the non-overlapping contact constraint between colliding bodies can be expressed in the following complementarity condition:

$$0 \leq v_n \perp \lambda_n \geq 0 \tag{1}$$

where v_n is a relative velocity along the normal direction at the contacting point where $v_n > 0$ implies separating, and λ_n is a (normal) contact force where $\lambda_n > 0$ implies pushing between each other. The meaning of the complementarity condition (1) is that the pushing force ($\lambda_n > 0$) should be generated only when the bodies become closer ($v_n < 0$), otherwise the force is zero ($\lambda_n = 0$).

Now, to calculate contact force $\lambda_{n,k}$ in discrete-time simulation of k-th step, discrete-time dynamics should be considered. For example, by using the passive midpoint integrator (PMI, [13], [14]), single rigid body dynamics with multi-point contact can be written as

$$M\frac{V_{k+1}-V_k}{T_k} + C_k(\omega_k)\frac{V_{k+1}+V_k}{2} = \sum_{i=1}^{n_c} J_k^{i,T} f_k^i \tag{2}$$

where $V_k := [v_k; \omega_k] \in \Re^6$ is a linear and angular velocity, $M := \mathrm{diag}[mI_{3\times 3}, J] \in \Re^{6\times 6}$ is a inertia matrix, $C_k := \mathrm{diag}[0_{3\times 3}, -S(J\omega_k)] \in \Re^{6\times 6}$ with $C_k = -C_k^T$ is Coriolis-like matrix, $J_k^i \in \Re^{3\times 6}$ is i-th contact Jacobian, and $f_k^i := n_k^i \lambda_{n,k}^i \in \Re^3$ is i-th contact force along its normal direction $n_k^i \in \Re^3$ with n_c being the number of contacts. With (2), the i-th normal direction velocity $v_{n,k+1}^i := n_k^{i,T} J_k^i V_{k+1}$ is typically employed to formulate (1) in discrete-time, where n_k^i, J_k^i, rather than n_{k+1}^i, J_{k+1}^i, are used to calculate $\lambda_{n,k}^i$ without iterative calculation while assuming that the configuration change of the rigid body is negligible during the contact instance.

The standard way to calculate $\lambda_{n,k}^i$ with (1)-(2) is formulating it as the *linear complementarity problem* [32]:

$$\bar{v}_{n,k+1} = A_{n,k}\bar{\lambda}_{n,k} + b_{n,k} \tag{3}$$

$$\bar{v}_{n,k+1} \geq 0,\ \bar{\lambda}_{n,k} \geq 0,\ \bar{v}_{n,k+1} \cdot \bar{\lambda}_{n,k} = 0 \tag{4}$$

where $\bar{v}_{n,k+1} := [v_{n,k+1}^1; v_{n,k+1}^2; \cdots; v_{n,k+1}^{n_c}]$ and $\bar{\lambda}_{n,k} := [\lambda_{n,k}^1; \lambda_{n,k}^2; \cdots; \lambda_{n,k}^{n_c}]$. This linear complementarity problem for only normal contact can be solved with a pivoting method (e.g., Lemke's algorithm [20]) or an iterative method (e.g., Projected Gauss-Seidel (PGS) method [33]) within polynomial time or exponential time in worst-case. However, solving frictional contact with the LCP is in general known as NP-hard problem [32], which becomes much difficult to solve with large number of contact and, thus, requires reducing the number of contacts.

B. Multi-Point Contact and Clustering

A large number of contact points can be produced when fine mesh models are used and/or the contact geometry is complex. However, reducing the number of contact points (i.e., clustering) is typically required, rather than using the large number of contact points as it is, for two reasons: 1) accelerating simulation speed especially for the real-time application; and 2) increasing numerical stability. When it comes to the numerical stability, large number of contact point typically makes the contacts overdetermined and produces ill-conditioned contact matrix $A_{n,k}$ in (3). Solving this ill-conditioned contact is difficult and often fails to find a solution. It can also slow down the convergence rate when the iterative solver is employed.

In this regards, the k-means clustering is widely employed for a simulation of multiple contacts between general mesh objects [22], [26]. Here, we briefly introduce the k-means clustering. The k-means clustering simply produces k clusters from n_c observations (contact points or normals) where the clustered number k is pre-defined by the user. It consists of three steps:

- Initialization: Choose k observations randomly and use them as means of each cluster.
- Assignment: Assign each observations to the nearest cluster. The euclidean distance between the mean and the observation is used.
- Update: Calculate the new means of the new cluster. Go to the assignment step until the clusters do not change.

One issue of this simple algorithm is that it can produce different resulting clustered contact points depending on the random initial guess. For a persisting contact, the change of contact points may produce chattering in the calculated contact forces. In this case, *warm starting*, where the previous step's clustering is used for the initial guess of the next clustering, can be employed to guarantee temporal coherence.

Our prime motivation is that this clustering affects not only the speed and numerical stability, but also the accuracy of the contact simulation. To our best knowledge, there is no research which analyses the effect of this clustering on the accuracy of contact simulation, since its analytical formulation is in most cases impossible. For this reason, we propose a novel data-driven learning-based contact clustering framework for the contact simulation with significant improvement of simulation accuracy as verified by real-world experiments.

III. Data-Driven Learning-Based Contact Clustering Framework

In this section, we introduce our data-driven learning-based contact clustering framework. As mentioned in Sec. I and illustrated in Fig. 2, the proposed clustering framework consists of two parts. The first part is the multilayer perceptron (MLP) network [34] which learns what contact points/normals should be activated for a given contact situation. Here we consider this as the classification problem,

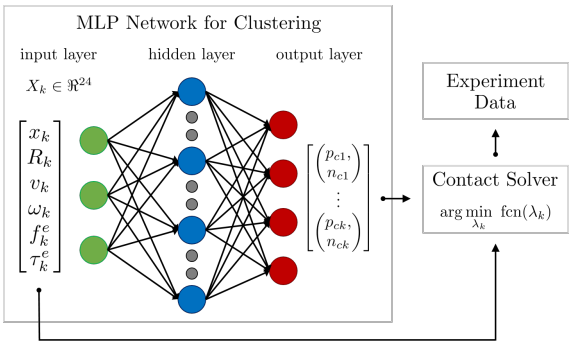

Fig. 2. Architecture of our data-driven learning-based contact clustering: MLP network for clustering and constraint-based contact solver.

where the network output is $\Re^{n_c}$ vector with its component meaning whether the contact is activated or not, among all possible contact grid (e.g., $n_c = 17$ for the surface contact in Fig. 3 left). We then choose the input X_k of this MLP network as the combination of positions, orientations, linear/angular velocities, and external forces and moments where the angular velocity and external moment are defined in its body frame (i.e., $X_k := [x_k; R_k; v_k; \omega_k; f_k^e; \tau_k^e] \in \Re^{24}$ for a single peg colliding to a static plate[1]). Whereas general contact clustering methods only consider the geometric information such as contact points, normals, or penetration depths ([22], [26]), here we employ this dynamics-related (i.e., including force and moment) input X_k, since it is indeed physically reasonable to infer that the clustered contact changes depending on both position, velocity, and force. Note that, the detailed network structure is to be defined for each contact scenario while its learning is done from the optimal contact points/normals for a specific contact instance which will be obtained by the second part in below. For example, the structure of the MLP network for the target contact scenarios of this paper (i.e., rather simple surface and edge contacts as in Fig. 3), is one hidden layer with its size being 40. Our experience indicates that the depth of the network should not be so deep for these cases to prevent over-fitting problem arising with 4 or 5 hidden layers.

The second part of the proposed contact clustering framework is the constraint-based optimization contact solver. This contact solver is utilized to find the optimal contact points and/or normals for the MLP network, which best matches the experimental data, while computing the contact force from the dynamics information X_k for all possible contact points and normals. For this, we reformulate (3)-(4) with a Coulomb friction as an optimization problem [25]:

$$\arg\min_{\lambda_k} \left(\begin{bmatrix} v'_{n,k+1} \\ v_{t,k+1} \end{bmatrix}^T \bar{A}_k^{-1} \begin{bmatrix} v'_{n,k+1} \\ v_{t,k+1} \end{bmatrix} + \begin{bmatrix} \lambda_{n,k} \\ \lambda_{t,k} \end{bmatrix} R \begin{bmatrix} \lambda_{n,k} \\ \lambda_{t,k} \end{bmatrix} \right) \quad (5)$$

$$\text{subj.} \quad v'_{n,k+1} \geq 0, \ \lambda_{n,k} \geq 0, \ \mu|\lambda_{n,k}| \geq |\lambda_{t,k}| \quad (6)$$

where $v'_{n,k+1} := v_{n,k+1} + \varepsilon v_{n,k} \in \Re^{n_c}$ is the desired normal direction velocity after contact with $0 \leq \varepsilon \leq 1$ being the

[1]Here, we slightly abuse the notation R_k to indicate vec(R_k), which is a stacking of each column of R_k, for brevity.

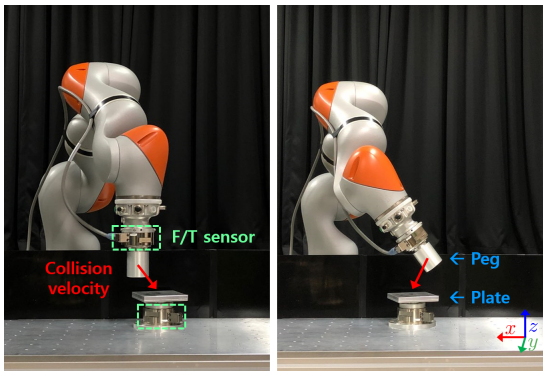

Fig. 3. Experiment setup consists of one KUKA robot manipulator, two ATI gamma sensors, and aluminium peg and plate. Vertically placed peg collides to the plate with different velocity for the first contact scenario (left) and inclined peg collides with different velocity for the second contact scenario (right).

restitution coefficient, $v_{t,k+1} \in \Re^{2n_c}$ is the tangential velocity, and $\lambda_{t,k} \in \Re^{2n_c}$ is the friction with $\mu \geq 0$ being the friction coefficient. The $\bar{A}_k \in \Re^{3n_c \times 3n_c}$ is the similar contact matrix in (3) where friction parts is added, and the $R := \text{diag}[r_1, r_2, \cdots] \in \Re^{3n_c \times 3n_c}$ is the positive regularizing matrix. The physical meaning of (5) is that, it maximizes the dissipation due to the contact while also minimizing the norm of total contact forces by R. Then, the solution of (5) can be obtained by solving

$$(\bar{A}_k + R) \begin{bmatrix} \lambda_{n,k} \\ \lambda_{t,k} \end{bmatrix} + \begin{bmatrix} b_{n,k} + \varepsilon v_{n,k} \\ b_{t,k} \end{bmatrix} = 0 \tag{7}$$

with a PGS-like iterative method under the constraint (6), which projects the nonlinear Coulomb friction similar to that of [33]. Note that every parameters of this optimization can be calculated from the network input X_k and inertial parameters (i.e., m or J). Consequently, we train the MLP network from the X_k to the optimal contact points and/or normals obtained from the above. We use the standard cross-entropy loss function the the training.

For the contact data acquisition, we configure the experiment environment as shown in Fig. 3. The experiment environment consists of 7-DOF robot manipulator (KUKA LBR iiwa 14 R820), two ATI Gamma F/T sensors, and aluminium peg and (static) plate. The mass and inertia of aluminium peg are $m = 339.53$ [g] and $J = \text{diag}[182767.90, 182767.90, 106769.63]$ [g$\times$mm^2] where its center of mass is 34.54 [mm] away from the F/T sensor along the z-direction in upright posture. The two F/T sensors are attached to the robot end-effector and the ground to measure the external force acting on the peg and the contact force between the peg and the plate. Then the torque controlled robot manipulator make the peg colliding with the plate. To make the peg contact with the plate along the predefined path, we design the end-effector impedance control [35] with the control gains $K = [800, 800, 800, 150, 400, 50]$ and $B = [250, 250, 250, 15, 30, 15]$ for each $\{x, y, z\}$ translation ([N/m] or [Ns/m]) and rotation ([Nm/rad] or [Nms/rad]). After that, we design two contact scenarios. In the first contact scenario, we make the peg contact to the plate with its surface to produce multiple contact points (left of Fig. 3). To obtain various contact data for this surface contact geometry, we change the approaching velocity, i.e., the incidence angle from 45° to 135° with 10 uniform grids and the magnitude from 0.02 to 0.07 [m/s] with 10 uniform grids. Thus, we obtain contact data for one hundred contact cases in this scenario. In the second scenario, we make the peg contact with its corner to the plate (right of Fig. 3). In this case, we change not only the collision velocity but also the attitude of the peg from 135° to 175° rotated along the y-axis. Since we used 5 grid points for this second scenario, total 125 contact data was obtained (i.e., 5 for incidence angle, 5 for magnitude, and 5 for peg attitude).

IV. Performance Evaluation and Comparison

A. Performance Evaluation of Data-Driven Clustering

In this section, we present the performance of our data-driven contact clustering. We plot the resulting force of (5)-(7) overlaid with the measured contact force from the experimental setup. The input X_k is directly calculated by using manipulator's joint angle and velocity, and the F/T sensor attached to the peg. We manually measure the friction coefficient $\mu = 0.15$ while the restitution coefficient $\varepsilon = 0.01$ being empirically estimated to make the contact behavior as close as possible to the experiment data. We also compare the contact force for the same input X_k obtained by (5)-(7) with the standard k-means clustering and the k-means clustering with warm starting.

For the first surface contact scenario, we take 17 contact points as the output of the MLP network. In this case, since both contact spaces of the peg and the plate are plain, we take all the normal of each contact point as z-direction (i.e, $n_k^i = [0; 0; 1]$). We also take three contact points as the output contact points since three points is the minimum number of points to define surface. Then, we find optimal contact point pairs by solving the calculated force of (5)-(7) and comparing it with the experiment data. We use 70% of total contact data to train the network and use the rest 30% data to test the resulting contact force.

The performance of each contact simulation for surface contact scenario is shown in Fig. 4. For readability, we shift the starting time of each force data as 5 seconds and we overlay the experiment data on the magnitude plot of each simulation (fourth plot) for more better comparison. It is shown that, with the proposed data-driven contact clustering, the accuracy of the calculated contact force increases. In particular, the normal direction force (third plot) is the same as the experiment. On the other hand, with the only standard k-means clustering, there is errors not only in the friction, but also in the normal direction. In addition, it is shown that the output contact force significantly vibrates if warm starting is not used. After all, the root-mean-square errors (RMSEs) of $(|F_k| - |F_k^e|)/|F_k^e|$ on the static contact of these three cases (i.e., MLC, KMC, KMCW) are 2.3%, 13.8%, and 13.1% respectively. Here we compare the static contact since its accurate generation is the very basic requirement for any simulators and also can be properly measured by the

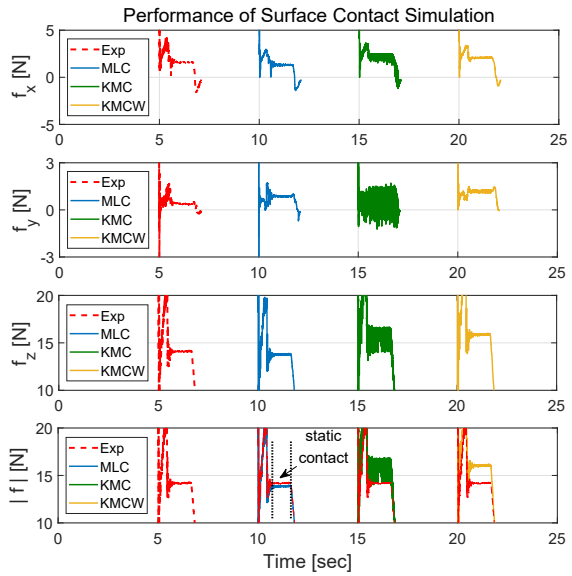

Fig. 4. Surface contact simulation performance: contact force of (5)-(7) with the machine learning-based clustering (w/ MLC, blue), with the k-means clustering (w/ KMC, green), and with the k-means clustering and warm starting (w/ KMCW, yellow).

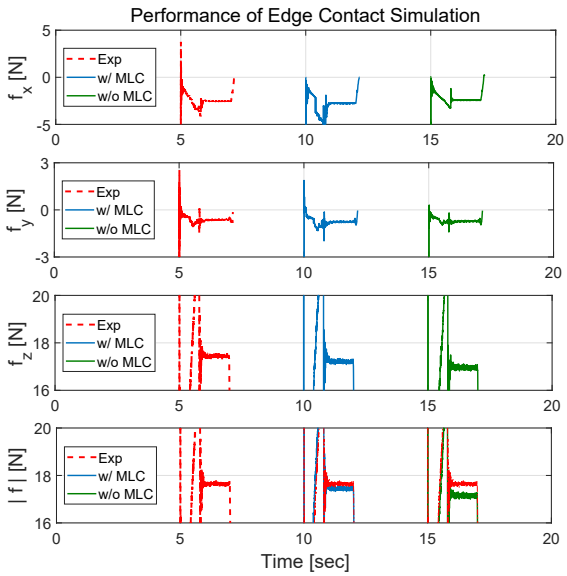

Fig. 5. Edge contact simulation performance: contact force of (5)-(7) with the machine learning-based clustering (w/ MLC, blue), without machine learning-based clustering (w/o MLC, green).

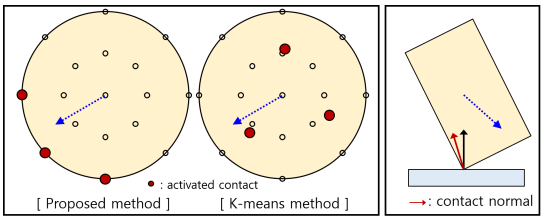

Fig. 6. Illustrative example of learning-based clustering: learning-based clustering produces different contact points compared to standard k-means clustering (left) or normal (right) depending on X_k (e.g, external force and colliding velocity).

F/T sensors with limited bandwidth (against impacts, noise, etc.). In fact, it is truly hard to accurately measure the impact data with F/T sensor, yet the tendency is similar with the dynamic contact (about 11% RMSE for the proposed method and about 24% RMSE for the k-means method).

In Fig. 6, we illustrate contact points clustered by the proposed method and by the k-means clustering. It turns out that the network choose different contact points depending on the colliding velocity (blue arrow in Fig. 6). This shows that our proposed clustering method, which fully exploit the dynamics-related information (i.e., velocity and external force etc.), can significantly increase the accuracy of multi-point contact simulation. Note that, we can increase the accuracy by gathering more data and/or employing normal direction clustering. In fact, we can further achieve performance improvement (2.3% $\rightarrow$ 0.8%) by gathering 100 more contact data for which we change the rotation of the plate to incorporate relative inclination between the plate and robot ground.

For the second edge contact scenario, we fix the contact point since it will produce almost single contact point between edge of cylinder and plain. However, we divide contact normals into seven grids of $\pm 45^\circ$ from the z-axis along the x- and y-axes (consequently 49 grid normals). Although it is easy to take $n_k^i = [0;0;1]$ from the plate, there exists unmodeled contact geometry such as chamfer, filleted-corner, or rough of surface etc. This normal grid can reflect such unmodeled geometry for the contact solver (5)-(7). In Fig. 5, we present the resulting contact force. As a comparison, we also plot the contact force calculated with $n_k^i = [0;0;1]$ (simply obtained from the geometry) for the same contact point. The results show that the MLP network actives $n_k^i = [0;0;1]$ as the contact normals in many cases, but there are still many other normal direction which shows better contact performance. See Fig. 6 left for the illustrative contact normal activation during edge contact. For this contact, the RMSEs with and without the learning-based clustering are 2.0% and 2.9% respectively. Although both contact simulations shows quite accurate performance for this case, it is obvious that the proposed method can increase the accuracy of contact simulation.

B. Contact Performance of Open-Source Robot Simulators

We also compare the performance with the existing open-source robot simulator V-REP [36] while changing its physics engine: Bullet, ODE, and Vortex. Each physics engine is widely used for robot simulation (e.g., bullet for Roboschool with OpenAI Gym, ODE for CHAI3D, and Vortex for many industrial training simulators). To compare them in identical environment, we use the remote API of the V-REP to control the virtual robot with C++ controller which is exactly same for the one used for the real KUKA robot. The configuration of V-REP simulator with the remote API is illustrated in Fig. 8.

To make both virtual and real KUKA robot as same as possible, we fine-tune kinematic parameters (e.g., joint length and its relative position) and dynamics parameters (e.g.,

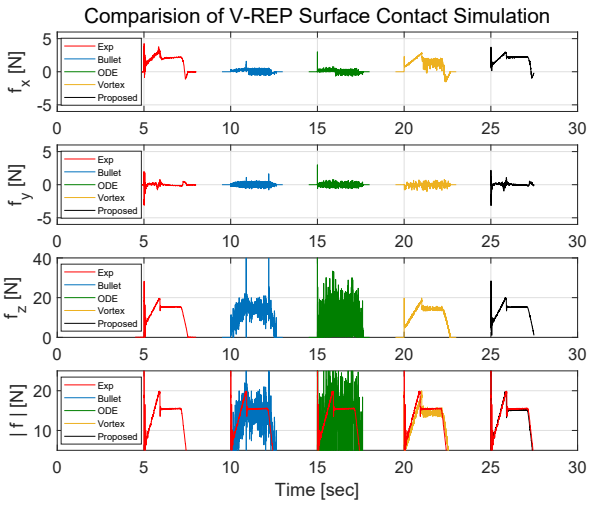

Fig. 7. Comparison of surface contact simulations done by V-REP and experiment.

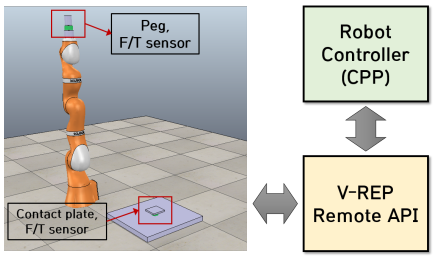

Fig. 8. Configuration of V-REP robot simulation. Same controller for real KUKA robot is used to control the virtual KUKA robot via remote API.

mass, inertia, and position of center of mass) of simulators. For the dynamics parameters, we use the identification of [37]. On the other hand, it is hard to stably simulate large control gain (i.e., K and B in Sec. III) in real-time unless using special simulation such as [14]. For this reason, we set $T_k = 0.2$ [ms] for the simulation time interval of V-REP, while the robot controller runs at 1 [kHz] same for the real KUKA robots. Note that we do not use Newton physics engine of V-REP since it always becomes unstable with the given configuration. In Fig. 7 and Fig. 9, we plot contact force data of the surface and the edge contact scenarios respectively, in order of experiment, Bullet, ODE, Vortex, and proposed method. As in Sec. IV-A, we shift the starting time of each force data as 5 seconds for readability. Although there is some vibration, Vortex shows the best performance among V-REP physics engine. Other Bullet and ODE exhibit significant chattering in the force data even though their graphical behaviors look similar to the naked eye.

Finally, we summarise RMSEs during the static contact of V-REP simulators and the proposed method in Table. I. The Vortex shows the best performance among V-REP physics engine (7.8% and 6.7% RMSE for both scenarios). However, our proposed data-driven contact clustering can produce more accurate contact force than Vortex physics engine.

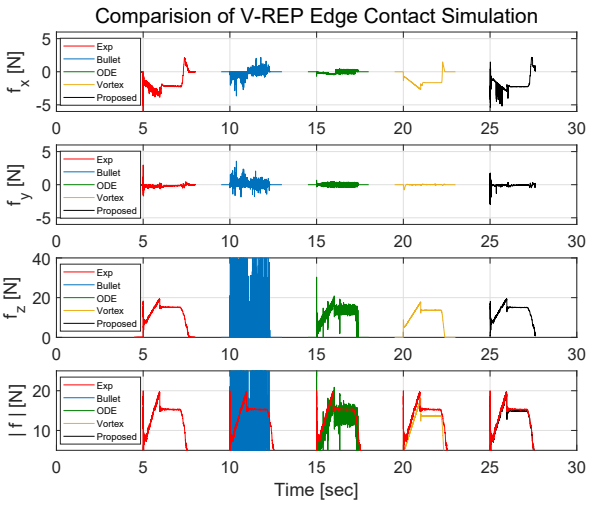

Fig. 9. Comparison of edge contact simulations done by V-REP and experiment.

Case	Bullet	ODE	Vortex	Proposed
Surface contact	10.4%	27.6%	7.8%	2.3% (0.8%)
Edge contact	78.5%	10.7%	6.7%	2.0%

TABLE I

PERFORMANCE EVALUATION AND COMPARISON OF V-REP SIMULATORS AND THE PROPOSED CONTACT CLUSTERING.

V. CONCLUSIONS

From the observation that contact clustering is not only crucial for numerical stability but also the accuracy of the contact simulation, we present the data-driven learning-based contact clustering framework for robot simulation which consists of the multilayer perceptron (MLP) network and constraint-based optimization contact solver. The MLP network is designed to classify the active contact points and/or normals from the given contact set. We train this MLP network by using the real-world experiment data and the contact solver employed for the dynamics-related input (e.g., position, velocity, and external force etc.). Performance evaluation and comparison is done for the proposed method with respect to the standard k-means clustering and V-REP simulators with Bullet, ODE, and Vortex physics engine. It shows that the proposed clustering method significantly outperforms the standard k-menas (with WS) and physics engine of Bullet and ODE, whereas show meaningful accuracy improvement over Vortex.

Our future research will include: 1) further extending the proposed method for more complex contact scenario such as peg-in-hole assembly, and 2) generalizing it, for example, by incorporating collision detection information.

REFERENCES

[1] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of the Conference on Computer Graphics and Interactive Techniques*, pages 23–34, 1994.

[2] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Int'l Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[3] M. B. Cline and D. K. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, volume 3, pages 3744–3751, 2003.

[4] J. Williams, Y. Lu, and J. C. Trinkle. A geometrically exact contact model for polytopes in multirigid-body simulation. *Journal of Computational and Nonlinear Dynamics*, 12(2):021001, 2017.

[5] N. Fazeli, E. Donlon, E. Drumwright, and A. Rodriguez. Empirical evaluation of common contact models for planar impact. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 3418–3425, 2017.

[6] J. Hwangbo, J. Lee, and M. Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

[7] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke1. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 4243–4250, 2018.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[9] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the HRP-2 humanoid. In *Proc. of IEEE/RSJ Intl Conf. on Intelligent Robots & Systems*, pages 3346–3351, 2015.

[10] E. Johns, S. Leutenegger, and A. J. Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *Proc. of IEEE/RSJ Intl Conf. on Intelligent Robots & Systems*, pages 4461–4468, 2016.

[11] J. M. Brown and J. E. Colgate. Minimum mass for haptic display simulations. In *ASME Dynamic Systems & Control Division*, pages 249–256, 1998.

[12] H. Courtecuisse, Y. Adagolodjo, H. Delingette, and C. Duriez. Haptic rendering of hyperelastic models with friction. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots & Systems*, pages 591–596, 2015.

[13] M. Kim, J. Kim, Y. Lee, and D. J. Lee. On the passivity of mechanical integrators in haptic rendering. In *Proc. of IEEE Int'l Conf. on Robotics & Automation*, pages 446–452, 2017.

[14] M. Kim, Y. Lee, Y. Lee, and D. J. Lee. Haptic rendering and interactive simulation using passive midpoint integration. *International Journal of Robotics Research*, 36(12):1341–1362, 2017.

[15] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[16] X. B. Peng, M. Andrychowicz, W. Zaremba, and Abbeel P. Sim-to-real transfer of robotic control with dynamics randomization. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 3803–3810, 2018.

[17] M. Tang, R. Tong, Z. Wang, and D. Manocha. Fast and exact continuous collision detection with bernstein sign classification. *ACM Transactions on Graphics*, 33(6):186, 2014.

[18] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. In *Computer Graphics Forum*, volume 21, pages 279–287, 2002.

[19] D. Baraff. Linear-time dynamics using lagrange multipliers. In *Proc. of the Conference on Computer Graphics and Interactive Techniques*, pages 137–146. ACM, 1996.

[20] J. E. Lloyd. Fast implementation of lemke's algorithm for rigid body contact simulation. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 4538–4543, 2005.

[21] J. Bender, M. Müller, M. A. Otaduy, and M. Teschner. Position-based methods for the simulation of solid objects in computer graphics. In *Eurographics (STARs)*, pages 1–22, 2013.

[22] A. Rocchi, B. Ames, Z. Li, and K. Hauser. Stable simulation of underactuated compliant hands. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 4938–4944, 2016.

[23] ODE. Open dynamics engine, 2001.

[24] E. Drumwright and D. A. Shell. Modeling contact friction and joint friction in dynamic robotic simulation using the principle of maximum dissipation. In *Algorithmic Foundations of Robotics IX*, pages 249–266. Springer, 2010.

[25] E. Todorov. A convex, smooth and invertible contact model for trajectory optimization. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 1071–1076, 2011.

[26] M. A. Otaduy and M. C. Lin. A modular haptic rendering algorithm for stable and transparent 6-dof manipulation. *IEEE Transactions on Robotics*, 22(4):751–762, 2006.

[27] J. Barbič and D. L. James. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics*, 1(1):1–14, 2008.

[28] E. Drumwright. A fast and stable penalty method for rigid body simulation. *IEEE Transactions on Visualization & Computer Graphics*, 14(1):240–231, 2008.

[29] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[30] Gazebo. Gazebo simulator, 2017.

[31] Bullet. Bullet physics engine, 2007.

[32] R. W. Cottle, J.-S. Pang, and R. E. Stone. *The linear complementarity problem*, volume 60. Siam, 2009.

[33] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization & Computer graphics*, 12(1):36–47, 2006.

[34] S. S. Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:, 2009.

[35] M. W. Spong, S. Hutchinson, and M. Vidyasaga. *Robot modeling and control*. John Wiley & Sons, Hoboken, NJ, 2006.

[36] E. Rohmer, S. P. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *Proc. of IEEE/RSJ Intl Conf. on Intelligent Robots & Systems*, pages 1321–1326, 2013.

[37] Y. R. Stürz, L. M Affolter, and R. S. Smith. Parameter identification of the kuka lbr iiwa robot including constraints on physical feasibility. *IFAC-PapersOnLine*, 50(1):6863–6868, 2017.