A distributed approach to haptic simulation

A distributed approach to haptic simulation

Dan Norman[1[0000−0001−7041−9972]], William Harwin[1,2[0000−0002−3928−3381]], and Faustina Hwang[1[0000−0002−3243−3869]]

[1] University of Reading, RG6 6AY UK {d.j.norman}@pgr.reading.ac.uk
{w.s.harwin,f.hwang}@reading.ac.uk
[2] RACE UK Atomic Energy Authority, Culham Science Centre Abingdon UK

Abstract. The quality of the physical haptic interaction and the need to link the haptic device or devices to high quality computer simulations in a time critical way are two key problems in modern haptic rendering. Additionally, in large simulation environments, the need to update the dynamic state of every object is required, even if the objects are not involved in haptic feedback. This can result in a decreasing haptic update rate for increasing simulation complexity.
A possible way to address these conflicting requirements is to consider control structures that operate at differing loop times and consider issues such as stability in the context of these loop times. This paper therefore, outlines a flexible rendering architecture to manage the conflicting requirements of simulation quality and simulation speed across multiple devices.

Keywords: Distributed systems · Haptics · Multi-finger · Bi-manual.

1 Introduction

One hurdle towards securing sustained interest for higher quality devices is the high cost of the haptic device and difficulty when adapting bespoke haptic devices to a specific task. Considering applications as a distribution of hardware and software may allow a more universal consideration of haptic based simulators and could also lead to the establishment of standards for haptic simulators.

This paper considers the above dilemma from the perspective of haptics as a distributed system. Distributed systems can be applied to haptics in two ways, physical distribution and control distribution. Physical distribution of the haptic hardware is considered as a potential method to create re-configurable multi-point haptic devices. Control distribution refers to the process of creating control software that has capabilities ranging from allowing physical distribution to function to increasing simulation stability in complex environments by limiting the number of objects handled in the haptic update, and therefore reducing the delay in the system[3].

Distributed control is widely used in the process industry and research on swarm robotics[10], where each robot can be considered a node in the control network. However, the bandwidth of the communication channels tends to be a

limiting factor in distributed systems. Therefore, there have been attempts to reduce the amount of data needing to be sent. One example is a system that uses state estimators[15] to predict what the other nodes are outputting rather than having each node sending an update to over the whole network.

2 Physical distribution

Physical distribution refers to how the haptic devices in a multi device setup are connected to each other, and the hardware that runs the control software. One of the simplest types of physical distribution is a pair of devices that have been connected to the same controller board to allow for two point haptic interaction, for example a two finger grasp[2].

The inputs and outputs of both devices are sent to the same controller board, which is connected to a computer and uses a control software library that expects two devices regardless of configuration. As such these two devices, which are approximating a single multi-point device can be reconfigured without needing any changes to the base level controller.

An alternative concept for implementing physical distribution is to create a system that consists of multiple device control boards connected by communication channels to the same simulation.

This can be used to create a system of devices with mixed locality, where some devices are connected directly to the machine running the simulation environment and others are connected indirectly. This concept has been partly implemented on a test-bed haptic environment developed at the University of Reading and is shown in fig. 1. The test-bed haptic environment (colloquially known as the Matrix) consists of two pairs of devices and their two corresponding control boards. Two control boards are used as each board can only have two devices connected to it. One of the boards is connected directly to the computer running the simulation while the other is connected indirectly via a secondary computer using an Ethernet cable. Communication over the Ethernet cable was handled by the communications protocol called UDP. The CHAI3D simulation framework[4], using the bullet physics module, is being used to test the effect of remote (indirectly connected) devices on simulation performance. The experiment consists of a pick and place task where the local pair represent one hand and the remote pair the other. The initial results from this experiment suggest that UDP over Ethernet could be a viable method for physical distribution, as the simulation remains stable and runs at at least 1kHz even when all four virtual end effectors are contacting the simulated cube and returning a force.

If the communication channels between the simulation (host) computer and the peripheral computers do not significantly increase the delay in the system it will not be necessary to have a device control board located within the computer running the simulation. The choice of communication channel and communication protocol types will impact system delay as each channel has constraints on capacity and speed. Additionally there is often a trade off between speed and reliability in communication protocols. For example, UDP type communication

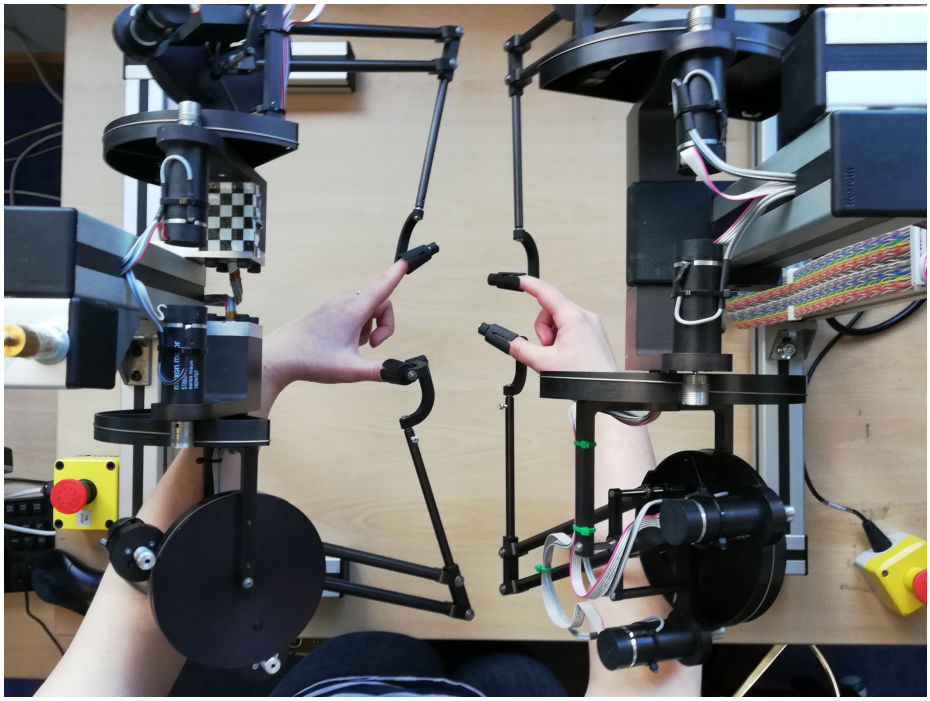

Fig. 1. The University of Reading bi-manual test-bed, comprising of 4 devices that are based on the phantom[7], arranged in pairs with one pair having a direct connection to the simulation environment and the other an indirect UDP based connection.

is fast with less computing overheads but receipt of the data is not guarantied, whereas TCP type communication is slower but guarantied to deliver data and packet order can be reconstructed[1].

Fig. 2 provides an example of a generalised configuration of the hardware where the devices are connected to a control board, which is within a peripheral computer/machine, with the encoder and motor values being sent via a communications channel. This configuration has the advantage of functioning as a plug-and-play type system where the number of devices is controlled by the number of peripheral machines. In addition, the peripheral machines can vary in complexity based off of whether they are intended to do any of the processing for the simulation or simply act as data relay stations.

A consideration when working with multiple nodes (e.g. multiple haptic devices, the virtual environment and other sensors such as the Vicon (vicon.com) for movement tracking) is the need to agree to a common coordinate frame. In the absence of a well defined relationship between these different coordinate frames, a least squares calibration can be used. Consider two sets of Cartesian data of the same point $\vec{s}$ moving in the workspace. If ${}^{f}S = \left[{}^{f}\vec{s}_1\ {}^{f}\vec{s}_2\ ...\right]$ is collected from the fiducial device while ${}^{c}S = \left[{}^{c}\vec{s}_1\ {}^{c}\vec{s}_2\ ...\right]$ represents the same positions but col-

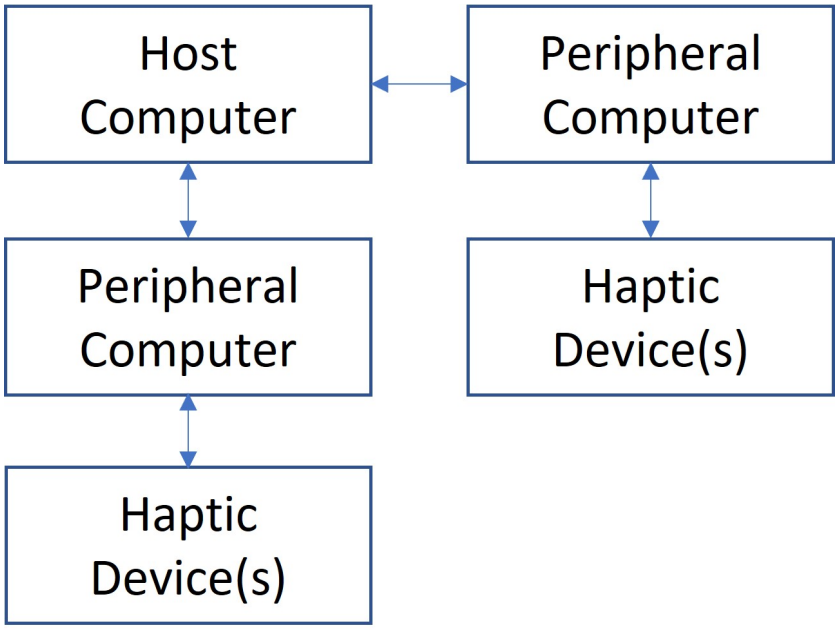

Fig. 2. Diagram displaying a generalised configuration of the hardware, where all of the devices are remote to the host computer, with communication channels denoted by the blue arrows. This configuration presents a plug-and-play type solution with peripheral machines of a lesser complexity than the host computer.

lected in the coordinate frame of the device to be calibrated then, in a linear workspace we can assume

$$^{c}S = {}^{c}_{f}T \; {}^{f}S$$

where ${}^{c}_{f}T$ is a 4×4 homogeneous transform matrix linking the two spaces. A least squares solution to this equation is straight forward, e.g. Moore-Penrose inverse. Leading to the solution

$${}^{c}_{f}T = ({}^{f}S^{T} \; {}^{f}S)^{-1} {}^{f}S^{T} \; {}^{c}S$$

This technique was used in the University of Reading test-bed using the Vicon as the fiducial device for calibration.

3 Control distribution

Control distribution can be considered from two perspectives: distribution of the physics simulation and distribution of the underlying computational process. Physics simulation distribution pertains to the question of how collision detection and physics update calculations should be implemented within the simulation code. Whereas, process distribution refers to the code functionality that allows

for physical distribution and the code that allows the distributed physics to interface with the rest of the program.

In haptics it is often assumed that having a physics engine that can handle all objects in the environment at a minimum update rate of 1kHz is a necessity. In practice haptic interactions can be stable at much lower update rates, and there are other factors that influence the limit cycles[11]. Currently there seems to be two main concepts of implementing physics in haptic simulations: First, having a single physics update loop that handles all objects in the simulation or second, having multiple physics update loops to enable graphical and haptic physics to update at different speeds. Chai3D[4] is an example of the first approach. An example of the second approach is Toia from Generic robotics[3], which uses a commercial physics engine 'Carbon' running on the multi-core CPU to compute the haptic physics, and the PhysX engine running on the GPU to update the physics of non-haptic objects at a speed suitable for stereoscopic computer graphics rendering[4].

Having multiple physics loops is likely to be preferable in complex multi-contact point haptic simulations since this will allow high loop times for individual haptic devices and complex physics on objects that are only needed for graphical rendering. The overhead for this approach is that the location of haptic objects must passed to the graphics render engine, and that it may be necessary to interchange objects between the haptics and graphics physics programmes. The benefit is that in large complex environments objects that don't need to be handled at haptic speeds can be handled in a separate "graphics" physics loop. Thus more haptic objects and points of contact can be computed in large and complex simulated environments before performance starts to degrade and instabilities start to appear.

3.1 Physics engines and haptics

The potential of using multiple physics engines in a simulation requires consideration into what physics engines are suitable for use in haptics and which are only suitable for the graphical rendering of non-haptic objects. The main consideration when choosing a physics engine for haptic, and robotic limb, simulation is the trade off between simulation speed and physical accuracy[5]. Table 1 lists some contemporary physics engine libraries with what type of dynamics they use, as well as their licence and primary application types.

Of the engines listed in table 1 most of them were created with the purpose simulating physics for computer games and as such are designed with more emphasis on ensuring the results create a visually pleasing result when graphically rendered, often at the cost of an accurate portrayal of the underlying physical system. This is because, of the three types of dynamics used in physics engines, force based dynamics are considered to be the slowest and most accurate

[3] https://www.genericrobotics.com/
[4] Information from personal correspondence

whereas position based dynamics are considered to be the fastest, but with the most deviation from the physical system it is simulating[8].

Impulse based dynamics are assumed to lie between force based dynamics and position based dynamics in terms of speed and accuracy. An approach to rendering both solids and fluids using a point based physics is also possible in a haptics context, but is computationally expensive requiring GPU hardware to achieve haptic rendering speeds[13].

Both ODE and Bullet are used by Chai3D[4], which is evidence that impulse based dynamics can be suitable for haptics though additional code needs to be generated by the haptic handler to calculate forces from outputs of the physics engines. Though the work by Erez et al[5] would suggest that MuJoCo would be better suited.

Table 1. Table to list a variety of Physics engine libraries and provide information useful for determining if they are suitable for haptics. Primary dynamics type is supplied when known explicitly, developers own terms used otherwise.

Engine	Dynamics	Licence	Primary applications	source
ODE	Impulse	open source	Gaming	ODE[5]
Havok	constraint based	commercial	Gaming	Havok[6]
PhysX	position	open source	Gaming	NVIDIA PhysX[7]
Box2D	Impulse	open source	Gaming	Box2D[8]
Bullet	Impulse	open source	Gaming and robotics	PyBullet[9]
Carbon	constraint based	commercial	gaming and animation	Numerion software[10]
MuJoCo	Impulse	open source	Robotics	MuJoCo[12]
Dart	constraint based	open source	Robotics	Dartsim[11]
Simbody	constraint based	open source	Bio-simulation and gaming	SimTK: Simbody[9]

However, regardless of the engine chosen to handle haptics, a second process is needed to handle the non haptic objects. This second process can either be the same physics engine as that used for computing haptic response to physical encounters, or like Toia it could be a different physics engine. If the object representation is different for the two engines then conversion between object representation would be needed across the functionality of objects rendered in these physics engines.

[5] https://www.ode.org/
[6] https://www.havok.com/havok-physics/
[7] https://developer.nvidia.com/physx-sdk
[8] https://box2d.org/
[9] https://pybullet.org/wordpress/
[10] https://www.numerion-software.com
[11] https://dartsim.github.io/index.html

3.2 Transferring objects between physics engines

A simulator that has the functionality of changing which of the physics engines an object is handled by is theorised to allow for further increases to the complexity of the simulation environment without decreasing haptic refresh rate to a point that causes instability.

The theory is based on the fact that objects that are not involved in haptics do not need to be computed as fast as objects that are involved in haptic feedback. Therefore, in order to keep the haptic refresh rate as high as possible only the objects currently involved in the calculation of the feedback force to be generated should be dynamically simulated at haptic speeds.

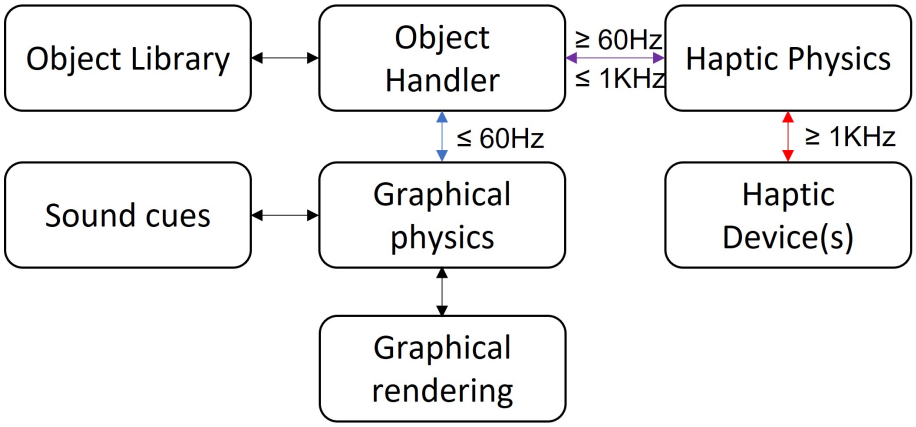

Fig. 3. Diagram displaying a configuration of the physics engines and object handler for a haptic simulator that can change which physics engine handles a particular object at each iteration. Object library contains all material, shape and location details at the beginning of the simulation. While the simulation is running these objects will be passed between the library and the local device haptic physics nodes. The object handler brokers these exchanges. This allows high speed update of objects that have a direct or semi direct contact with the individual haptic device. Two approximate channel capacities are shown. Blue channels are of the order of $60kHz$, red channels are notional haptic speeds of $> 1kHz$

In order to achieve a haptic simulator that can change which physics engine an object is handled by an object library that is accessible by both physics engines and an object handler function, that determines which engine has access to what objects, would be required.

Because the system would now be comprised of three elements, each with their own data and refresh rate requirements, the distribution of the control software can considered in terms of the capacity and speed of communication between the elements.

The Diagram in fig. 3 provides an example of how the two physics engines could interface with the object library via the handler and other elements necessary for simulation. It also illustrates the potential communication speeds between the elements, based off of the refresh rate of their internal loops.

As the object handler controls which of the objects are handled by the haptics physics it needs to be able to determine when to assign the objects. In addition the handler needs to make these assignments in regards to every connected haptic device and as such needs to happen at fast loop speeds, preferably the same speed as the haptic physics loop.

Therefore it is proposed that a method based on broad phase collision detection be used in the handler to determine which objects in the environment are to be handled by haptics physics. This is because broad phase collision detection methods are designed to operate quickly over all objects in the scene in order to ensure only objects that might be colliding are passed to the next phases of collision detection and resolution[6].

3.3 Multiple haptics physics engines approach

While Fig. 3 describes a system that uses two physics engines, one for graphical objects and one for haptic objects, it would be possible to expand the framework to work with more than two physics engines. When using three or more physics engines for example, one of them would handle the graphical physics and the others would handle the physics for a subset of objects across the set of haptic devices.

In such a system each device or device group can be considered to be part of an inner closed loop system (Fig. 4), within the larger system, with impedance provided by the human user and object information being exchanged with the object handler.

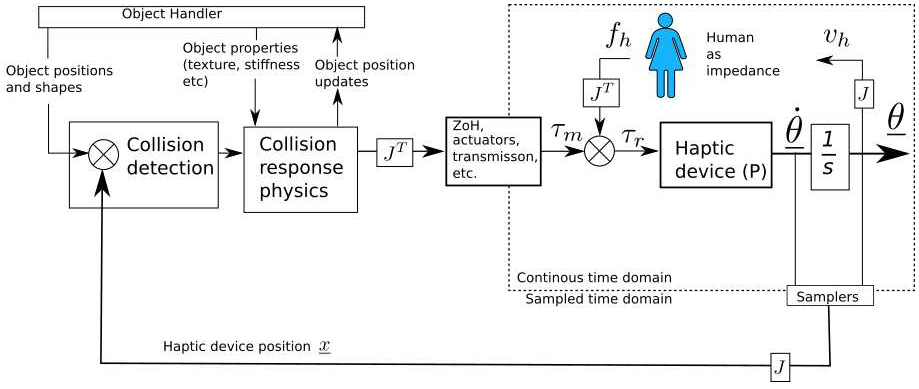

Fig. 4. Block diagram of haptic device showing a generalised inner closed loop with human interactions considered as an impedance and haptic physics and exchange of information with object handler.

Fig. 4 shows a possible node for one part of the arrangement described above. Stability of this type of structure is relatively well understood for simple physics. For example Colgate and Schenkel [3] show a stability condition for Fig. 4 where the collision detector is a 'relop' function and the collision response is a PD controller of the form

$$H(z) = K + B\frac{1 - z^{-1}}{T}$$

where K is the proportional gain of the digital controller, B is the differential gain and T is the sampling time of the control loop. The haptic device is assumed to have an implicit damping b in which case Colgate and Schenkel stability requires

$$b > \frac{KT}{2} + |B|$$

Additionally, the variables J and J^T in the control loop are generalised transforms from Cartesian space to joint space and back again, and are not necessarily Jacobian matrices.

This type of controller assumes a backdrivable type of haptic device. Devices with an admittance control allow the forces from the human f_h to be included in the control loop. The controller is necessarily a mixture of continuous time elements (amplifiers, actuators, linkages) and discrete time elements (digital computer with sampler and zero order holds (ZoH)).

Because the Object handler as well as physics engines on other nodes will have a lower update time contact instability will be possible. However, passing the control of an object to a haptic device that is local to the persons space should enable these instabilities to be better managed. This would also lend itself to collaborative simulations where the two users can come into contact with the same objects or each other. This is due to the fact that until the users come into close proximity, or indirect contact, their respective devices are not impacted by the feedback forces caused by the other user. Though when in close proximity or contact, the two physics loops will either need to communicate or be superseded by another physics loop that is included to handle these situations.

Additional benefits of a system that uses multiple haptic physics loops are that it would be tolerant of individual node failures since all haptic devices run their own version of the physics, and that the devices being connected and their physics engine do not have to be of the same type. For example, a simulation could be run where one of the devices is a device like the phantom[7] and the other could be a planar device like the haply 2diy[14].

4 Conclusions and further work

This research sought to determine if viewing haptics simulation in terms of a distributed system would have the potential to improve performance by addressing the problems of requiring high quality physical haptic interaction in complex environments and the need to link devices to high quality computer simulations

in a time critical way. This was done by considering distribution in terms of physical and control distribution.

The work on physical distribution has shown that stable physical distribution can be achieved if the communication channels used have the required bandwidth and speed. Initial experiments using the the university of Reading haptic testbed (Fig. 1) have suggested that the UDP type communication protocol using Ethernet cables are an example of a suitable type of communication channel and protocol for producing stable simulations. However, more work needs to be done to determine the best choice for communication channel and protocol.

The work on control distribution has shown how expanding on the type of control distribution used in Toia allows for the system to be considered a closed loop with many inner closed loops. Which in turn allows for a system to be proposed that takes Toia's approach to reducing computation at haptic speeds even further (Fig. 3), by varying the number of objects in the haptic physics every loop.

Following on from this research, work focusing on developing the object handler functionality, it's associated simulation control structure (Fig. 3) and testing its performance in large, complex, and collaborative environments to determine if it does provide a better alternative to a single powerful physics engine and computer running all the haptic objects will be carried out.

References

1. AL-Dhief, F.T., Sabri, N., Latiff, N.A., Malik, N., Abbas, M., Albader, A., Mohammed, M.A., AL-Haddad, R.N., Salman, Y.D., Khanapi, M., et al.: Performance comparison between tcp and udp protocols in different simulation scenarios. International Journal of Engineering & Technology **7**(4.36), 172–176 (2018)
2. Barrow, A., Harwin, W.: Design and analysis of a haptic device design for large and fast movements. Machines **4**(1), 8 (2016)
3. Colgate, J., Schenkel, G.: Passivity of a class of sampled-data systems: Application to haptic interfaces. Journal of robotic systems **14**(1), 37–47 (1997)
4. Conti, F., Barbagli, F., Balaniuk, R., Halg, M., Lu, C., Morris, D., Sentis, L., Warren, J., Khatib, O., Salisbury, K.: The chai libraries. In: Proceedings of Eurohaptics 2003. pp. 496–500. Dublin, Ireland (2003)
5. Erez, T., Tassa, Y., Todorov, E.: Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In: 2015 IEEE international conference on robotics and automation (ICRA). pp. 4397–4404. IEEE (2015)
6. Luque, R.G., Comba, J.L., Freitas, C.M.: Broad-phase collision detection using semi-adjusting bsp-trees. In: Proceedings of the 2005 symposium on Interactive 3D graphics and games. pp. 179–186 (2005)
7. Massie, T.H., Salisbury, J.K., et al.: The phantom haptic interface: A device for probing virtual objects. In: Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems. vol. 55, pp. 295–300. Citeseer (1994)
8. Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. Journal of Visual Communication and Image Representation **18**(2), 109–118 (2007), https://www.sciencedirect.com/science/article/pii/S1047320307000065

9. Sherman, M.A., Seth, A., Delp, S.L.: Simbody: multibody dynamics for biomedical research. Procedia IUTAM **2**, 241–261 (2011), https://www.sciencedirect.com/science/article/pii/S2210983811000241, iUTAM Symposium on Human Body Dynamics
10. Spears, W.M., Spears, D.F., Hamann, J.C., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous robots **17**(2), 137–162 (2004)
11. Swaidani, L., Steele, L., Harwin, W.: Motor shaft vibrations may have a negative effect on ability to implement a stiff haptic wall. In: International Conference on Human Haptic Sensing and Touch Enabled Computer Applications. pp. 252–263. Springer (2018), http://www.cybernetia.co.uk/pubs/paper-1229.pdf
12. Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033. IEEE (2012)
13. Tse, B., Barrow, A., Quinn, B., Harwin, W.S.: A smoothed particle hydrodynamics algorithm for haptic rendering of dental filling materials. In: 2015 IEEE World Haptics Conference (WHC). pp. 321–326. IEEE (2015)
14. Weill-Duflos, A., Ong, N., Desourdy, F., Delbos, B., Ding, S., Gallacher, C.: Haply 2diy: An accessible haptic plateform suitable for remote learning. In: Proceedings of the 2021 International Conference on Multimodal Interaction. pp. 839–840 (2021)
15. Yook, J., Tilbury, D., Soparkar, N.: Trading computation for bandwidth: reducing communication in distributed control systems using state estimators. IEEE Transactions on Control Systems Technology **10**(4), 503–518 (2002)