PrendoSim: Proxy-Hand-Based Robot Grasp Generator

Keywords: Robot grasping, simulation, virtual environment, virtual reality.

Abstract: The synthesis of realistic robot grasps in a simulated environment is pivotal in generating datasets that support sim-to-real transfer learning. In a step toward achieving this goal, we propose PrendoSim, an open-source grasp generator based on a proxy-hand simulation that employs NVIDIA's physics engine (PhysX) and the recently released articulated-body objects developed by Unity. We present the implementation details, the method used to generate grasps, the approach to operationally evaluate stability of the generated grasps, and examples of grasps obtained with two different grippers (a parallel jaw gripper and a three-finger hand) grasping three objects selected from the YCB dataset (hammer, screwdriver, and scissors). Compared to simulators proposed in the literature, PrendoSim balances grasp realism and ease of use, displaying an intuitive interface and enabling the user to produce a large and varied dataset of stable grasps.

1 INTRODUCTION

Robots exhibit good manipulation skills in structured environments, where objects are in controlled positions, information about the environment is complete and accurate, and actions are repetitive. However, grasping and manipulation become challenging in unstructured environments, such as households and hospitals. Here, perception can be noisy and unreliable, so each action requires reasoning on imperfect sensory data. Despite these problems, robots are increasingly used in industry settings that require complicated dexterous manipulation, such as grasping a hammer to pound a nail. Sometimes, robots are used in collaborative, joint-manipulation tasks with human partners, *e.g.*, during the assembly of a piece of furniture, where the robot could pass tools to the human partner (Ortenzi et al., 2020b) or directly tighten a screw on a wood panel held by the human partner. Collaboration adds another level of complication, as there are two interacting agents sharing the space and the tools (Ajoudani et al., 2017).

To be useful in these scenarios, robots need to be able to grasp objects. Robot grasping has been studied for over three decades, and enormous advances have been achieved in perception, planning, and grasp synthesis. However, some of the assumptions on which most of the work is based are still difficult to relax: objects are usually assumed to be rigid, friction is generally ignored or assumed to be uniform for the entire object, functional parts are usually not accounted for, and the task to be performed with the object is not considered (Ortenzi et al., 2019).

Many current approaches to grasping are looking into robot learning as a way of overcoming these limitations (Levine et al., 2018). However, learning grasping policies applicable to a wide range of situations requires large datasets of grasps, *i.e.*, employing different objects, different poses for the objects, different grippers, and different tasks. It is generally impractical to obtain a high number of successful grasps with real robots; thus, simulators are commonly used instead. However, the main challenges with simulators are (i) to generate an extensive range of grasps and (ii) to ensure that the grasps sufficiently replicate the characteristics of real robots grasping real objects, minimising sim-to-real differences.

Figure 1: Spherical object grasped using PrendoSim's BarrettHand gripper (solid rendering) and depiction of the kinematic hand employed in the proxy-hand method (mesh lines).

Here, we present PrendoSim, an open-source grasp generator based on the popular Unity game engine (Unity Technologies, San Francisco, CA), which allows one to create visually and physically realistic grasping by using advanced physics simulation of dynamic properties, such as friction, weight, weight distribution, and inertia (Fig. 1). As opposed to

most other grasp generators, PrendoSim has been designed to require only minimal knowledge of mechanics and can be controlled with an intuitive user interface (UI) with no programming requirements. The generated grasp configurations are stored both as an image (PNG format) and in a standard JSON file format (JavaScript Object Notation) to enable visual and numerical analysis and sharing to hardware systems. The version of PrendoSim presented in this paper can be downloaded for free from the following URL: (removed for anonymisation)

2 RELATED WORK

The synthesis and evaluation of grasp candidates involve several aspects that make selecting a suitable grasp challenging, *e.g.*, the object's geometry, the object's friction parameters, and the mechanical characteristics of the gripper. Some approaches capitalise on the design of the hardware to ensure stable grasping and simplify control. This is the case for industrial applications where either suction or parallel jaw grippers are generally used (Honarpardaz et al., 2017). When more sophisticated grippers are employed, *e.g.*, multi-fingered hands inspired by the human hand, the increase in dexterity comes at the cost of higher complexity and control effort (Berceanu and Tarnita, 2012).

Given a gripper and an object to grasp, there are usually a number of possible grasps to choose from. Force- and form-closure methods select grasp candidates based on stability. For example, force-closure is based on a mathematical formula that analytically determines whether the gripper and the object form a system resistant to external wrenches (Nguyen, 1988). For this, physical quantities like the geometric configuration of the grasp and of the object, the forces applied, gravity, and friction all need to be considered to compute whether the grasp is stable (Ferrari and Canny, 1992; Bicchi and Kumar, 2000; Ding et al., 2001). However, there is no consensus on how to define stability across the community. More operational definitions consider grasps to be stable whenever the object is held by the robot gripper for more than a certain amount of time, or after shaking (Bekiroglu et al., 2020). PrendoSim similarly adopts an operational definition of stability, which we will cover in detail in the following sections.

There are a number of different approaches for grasp detection. Work like (Gualtieri et al., 2016) proposed to use artificial neural networks to generate and evaluate grasps for parallel grippers. At first, the network learns the key features for grasping objects from a dataset. Then, it picks the grasp configuration that is similar to the ones that had a greater degree of success. Such simulation-based approaches can be successful in simulated environments and with known objects, but they do not usually allow for successful sim-to-real transfer learning. Other methods propose employing local features to make the grasp more robust (Adjigble et al., 2018). In this case an algorithm scans the object surface to identify areas where the local object curvature matches the finger's curvature. This method yields more robust results as it is based on real point clouds and on the kinematics of the gripper, and it could extend to more complex grippers with more than two fingers. However, several factors (dynamics, forces, friction and the task context) are not considered in selecting and evaluating the grasps.

Although researchers are attempting to improve grasping generation strategies to include new elements such as friction coefficients (Nguyenle et al., 2021), only a few of these new grasp generators are open source and available for download. GraspIt! (Miller and Allen, 2004) is among the most frequently used simulators. It provides a framework for testing grasp strategies on a variety of hands. It has a 3D graphical interface, and both robots and obstacles can be loaded using Python scripts. URDF and XML configuration files allow the user to simulate complex scenes with more than one robot and object, *e.g.*, a kitchen with a service robot. The software generates a set of grasps that satisfy the force-closure metric and are feasible considering the robot kinematics. The physics engine of GraspIt! allows the user to pick up objects and check the dynamic behaviour of both robot and object. Because of its versatility, researchers have developed several packages that integrate GraspIt! with the Robot Operating System (ROS) and Gazebo, a popular visualization and development tool in the robotics community. However, the main branch of GraspIt! has been discontinued, and only a few volunteers are maintaining the software for specific applications. Also, GraspIt! is platform dependent and only available for the Linux operating system.

Another very common simulator is Simox (Vahrenkamp et al., 2012). Simox is a robotic toolbox for motion and grasp planning, and it allows the import of complex kinematic chains, like humanoids and mobile robots. Simox is developed in C++, which affords efficiency, and it can be used by either script or interface. An additional strength is that it is platform independent. The user may load the CAD model of an object and a hand using an XML file. Then, the algorithm generates a sequence of robust grasp points on

the object surface. A collection of metrics are already available in the software. Additionally, users may define their own metric. By default, Simox produces grasp locations according to the force-closure metric. This software considers friction by defining friction cones at the contact points between hand and object. However, it discounts the effects of an object's weight and distribution of mass while grasping. Thus, grasps produced by Simox may fail to keep the object in-hand in a real scenario due to mismatches between the simulated and real interactions. Moreover, Simox installation requires advanced programming skills to set up the environment correctly. Our PrendoSim eases the burden for the user with a friendly installation process suitable also for programming-naive users.

Beside specific software targeted at grasping, the robotics community also resorts to more general simulators, *e.g.*, CoppeliaSim[1], PyBullet[2], MoveIt[3] and MuJoKo[4]. These software packages offer well-rounded physics engines and allow the simulation of complex scenes, so that they can be employed to test grasps. However, they are not designed to generate grasps and do not implement any grasp generator by default. Therefore, users must code a generator that is tailored to the application at hand. Only a few programmers have released grasp generators for these software packages, *e.g.*, multi-contact-grasping[5] is a grasp generator for CoppeliaSim, and OpenAI developed a Python package for PyBullet.

Besides the custom grasp generators or more generic simulators mentioned so far, there is another class of simulation engines offered through game engines, such as Unity. Until recently, however, it was challenging to create realistic simulators in this class of software, because engines emphasised smooth game performance rather than physical realism. In other words, modelling kinematic chains or a system of connected hinges, as often seen in robotics, would have resulted in unstable and unrealistic motion, unfit for translating to a real robot. With the latest release of Unity v2020.1 beta, this situation has now changed. Major game engines like Unity are now incorporating state-of-the-art, hardware-accelerated, real-world physics simulators with the aim to support a growing robotics community in research and prototype development.

In the following sections we describe how our grasp generator takes advantage of Unity's new and powerful physics simulator, our proxy-hand-based grasping method, and the performance test to evaluate generated grasps. We report a number of resulting grasps and describe the output of PrendoSim.

[1]https://www.coppeliarobotics.com/

[2]https://pybullet.org/wordpress/

[3]https://moveit.ros.org/

[4]http://www.mujoco.org/

[5]https://github.com/mveres01/multi-contact-grasping

3 PRENDOSIM

Our simulator takes advantage of Unity's articulated bodies to define a set of connected objects organized in a hierarchical parent-child tree. This framework can be used to define mutually constrained parts, such as joints, digits and rotational limits in a robot gripper. The root of the hierarchy tree is the base of the robot gripper, and the furthest components in the tree are the tips of the gripper's digits. Moreover, we implement a new method of contact force estimation for simulated robotic object grasping. The estimated grip forces are checked in our grasp stability test, where we gradually increase the loading on the object until it slips.

Our grasp generator also takes advantage of Unity's physics materials to define friction. To compute the friction coefficient, which determines the ratio between lateral force and normal force, our approach combines material-dependent parameters of the gripper and object and a Coulomb friction model with separate static and kinetic coefficients. We describe the implementation details in Sect. 3.3. Finally, physics materials also define a term for elasticity, but in our simulations we have set this term to zero. In other words, we simulated only rigid surfaces even though elasticity could be varied to extend the scope of the project.

3.1 Proxy Hand

We adopted a method that is commonly employed in haptic rendering and virtual reality simulation. The method can take several names depending on the specific implementation such as proxy, avatar, virtual tool, or god-object.

The proxy-hand method (Borst and Indugula, 2005) has been shown to simulate physically accurate, grip-force-based object grasping. We apply this method using Unity's "Articulated Body" component. We animate a kinematic gripper, which is a copy of the kinetic gripper without any physical properties, so that it cannot physically interact with the virtual objects or the environment. The articulation component of each joint defines the associated stiffness and damping parameters, as well as the maximum force or torque the joint can exert and the joint's positional or angular limits. For each of the robot grippers used, we have set these values to match the manufacturer's specifications, using the recommended control

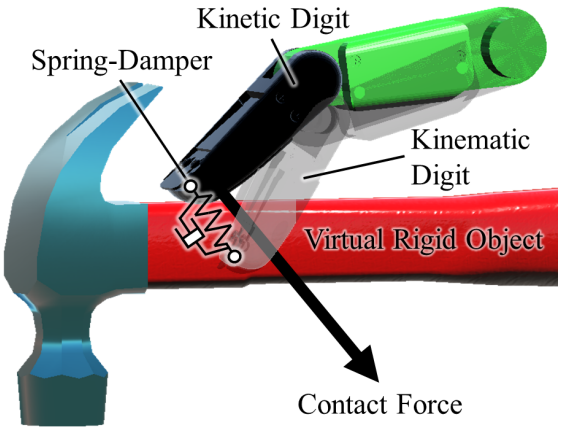

Figure 2: Contact force generation using the proxy-hand method. The positional discrepancy between the kinematic and kinetic digits results in a force ($\mathcal{F}$) computed from the spring-damper linking the two.

parameters for the proxy hand's stiffness and damping. The opening and closing motions of the gripper are driven gradually by updating the target angle or position. We adjust the target step size based on the joint limits. Without a graspable object, all joints will be driven to their angular or positional limit, reaching the gripper's closed state at the same time.

When grasping an object, we drive the joint targets until a certain moderate contact force threshold between the digits and the object has been reached. We compute this contact force by coupling the tips of each of the kinematic and kinetic gripper's digits with a spring and a damper (Fig. 2). Unity's default collision-detection algorithm maintains contact between the kinetic digits and the object, while the kinematic digits penetrate into the object. Using Hooke's law at steady state, the computed contact force is proportional to the distance between the tips of the kinematic and kinetic joints (penetration distance) (Höll et al., 2018). To achieve a moderately secure initial grasp that does not overly squeeze the object, we set the contact force threshold based on the known object mass. For example, we set a 5.5 N contact force threshold for an object with 500 g mass, which is about 10% larger than the minimum 5 N required to hold the object. This contact force is divided evenly between the digits in contact with the object so that the set total grip force is exerted on the object.

3.2 Grasp generation

Our simulator synthesises gripper configurations based on a grasp quality index computed through a dynamically changing load force, which also accounts for friction. The following steps estimate the performance of a randomly generated object grasp under friction and dynamics. This sequence of actions forms the core strategy of PrendoSim; it allows derivation and validation of every grasp candidate for any given target object.

1. The gripper starts in a fully open configuration. Any DOF available for changing the type of grasp is set to a random joint value.
2. One of the provided target objects is randomly chosen and instantiated in front of the gripper, with a random pose held statically for one second.
3. Within the one-second window the gripper proceeds to close as described in section 3.1.
4. The target object is then released to allow physical interaction with the gripper's digits.
5. The object will then either fall due to gravity or reach a stable state (*i.e.*, no motion for two seconds) within the gripper. In the former case, we discard the grasp. In the latter, we consider the grasp successful.
6. To further evaluate grasp stability, we gradually increase the mass of the object at a rate of 1.2 kg/s to measure the point at which the object starts to fall out of the gripper. An empirically selected velocity threshold of 2.5 cm/s is used to detect when the object starts to fall.
7. The object mass at this point is defined as the *critical mass*.
8. Optionally, this value, together with the gripper's joint configuration, applied contact forces and a screenshot of the scene are stored in a JSON-data and PNG-image file, respectively.

By repeatedly following this procedure, PrendoSim obtains a collection of successful grasps and provides a metric for the stability of each grasp based on the *critical mass*. This approach is agnostic to the gripper type and target object.

3.3 Grippers and Objects

PrendoSim provides two types of grippers and a set of three target objects with different shapes.

3.3.1 Grippers

The two built-in grippers consist of a two-finger, parallel jaw gripper (Franka Emika gripper[6]), and a three-digit gripper (BarrettHand BH8-series from Barrett Technology[7]); both appear in Fig. 3. Each

[6]https://github.com/frankaemika
[7]https://advanced.barrett.com/barretthand

Table 1: Static (π_s) and kinetic (π_k) friction parameters for the two grippers used in PrendoSim. Friction coefficients μ_s and μ_k are calculated by averaging the friction parameters of the gripper and the object surface it is touching (shown in Table 2).

Gripper	π_s	π_k
BarrettHand	0.70	0.25
Franka Emika	0.90	0.50

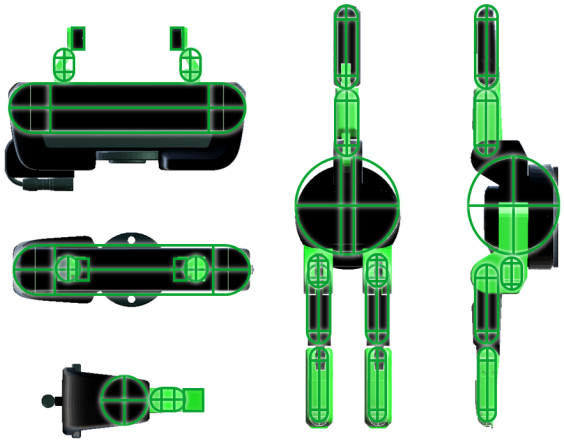

Figure 3: Franka Emika (left) and BarrettHand grippers (right) with highlighted primitive-shapes colliders (green lines).

gripper has a set of primitive shape colliders to define its physical shape. Unlike other simulators, we use primitive shape colliders for improved computational efficiency as our simulator runs in real time (up to 500 frames per second). Finally we equip each gripper with one physics material to describe the gripper's contact-point friction parameters (Table 1).

The choice of these two grippers allows for a large range of variability in the types of grasps that can be generated. Parallel jaw grippers are particularly favoured in industry as they offer a high degree of robustness, and only two opposite grasping points must be selected to pick up an object. However, more sophisticated behaviours, such as the distinction between precision and power grasps, are not possible. In contrast, a multi-finger hand allows for more flexibility in the choice of grasp. This higher complexity comes at a cost of generally lower grasp robustness and higher control effort.

The BarrettHand is a multi-fingered gripper with a higher degree of dexterity that can grasp objects of various sizes and shapes and at different orientations. The three articulated fingers are composed of two independently moving joints with ranges of 140° (finger base joint) and 48° (fingertip). They can each apply a maximum force of 20 N at the fingertip. Two of the fingers have a 180° spread (mirrored joint) that allows a change in grasp configuration, *e.g.*, from cylindrical grasp to spherical grasp. We obtain the open-hand configuration from the kinematic description of the BarrettHand when every finger is fully open. As for the closed-hand configuration, we first load the gripper into Unity. Then, we set a capsule collider for every digit of the gripper. The BarrettHand has three fingers with two phalanges for each finger; therefore, we obtain a total of six capsule colliders. Moreover, we set a collider for the base of the gripper.

PrendoSim contains two built-in grippers, but it is possible to include a new gripper by defining the open-hand configuration and the colliders for each of its digits, which will drive the kinematic gripper throughout the closure movement.

3.3.2 Objects

There are presently three built-in objects in PrendoSim: a hammer, a screwdriver, and a pair of scissors. We selected these models from the YCB dataset (Calli et al., 2017) because they are familiar to most people and offer different degrees of constraint difficulty for their intended use (Ortenzi et al., 2020a). We divided each object into two functional areas: a tool interface (grey area) and a grasp interface (red area) (Fig. 4) (Osiurak et al., 2017). We accounted for materials, weight distribution, and friction separately on both parts of each object (Table 2).

In particular, a hammer is designed to be grasped by the handle (graspable side) and used by the head (tool side). The handle is quite large, with respect to the entire body, and can be grasped at any point. The only constraint to its use is the head orientation with respect to the object to pound. Thus, a hammer offers loose constraints on the grasp.

The screwdriver is generally smaller than a hammer, but the two structures are similar. The body can be divided into two main functional areas: the handle and the metallic rod. The handle is the graspable part, and the metallic rod is the interface between the tool and the object to be tightened or loosened with the screwdriver. A screwdriver presents tighter constraints with respect to the hammer, as the orientation of the rod has to match the approaching direction of the screw, and the tip must be slightly inserted into the head of the screw.

Finally, the scissors present the most stringent constraints among the three tools. Although they have a handle like a hammer and a screwdriver, this handle requires the user to insert the fingers through the rings to achieve the correct cutting motion. The blades must enclose the material while an opening-and-closing motion of the fingers performs the cut.

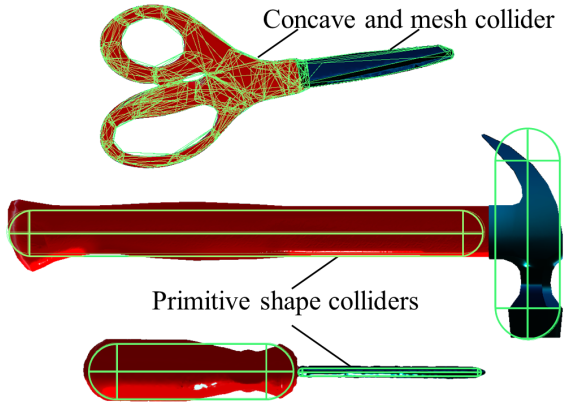

Figure 4: Geometry of the object colliders. Top: Scissors with a concave collider applied to the handle (graspable side) and a mesh collider applied to its tool side. Middle and bottom: Hammer and screwdriver with two capsule colliders each; a capsule for the tool side and another one for the graspable side. All colliders are shown as green lines.

Each object has a set of two specifically chosen colliders that conform to the object's shape as closely as possible (Fig. 4). For the scissors, we have chosen a concave mesh collider for the graspable side, which more accurately wraps around the handle, leaving the two holes as empty space. For the tool side of the scissors, we have chosen a mesh collider. As for the other two objects, a capsule collider was sufficient to define the objects for accurate grasping.

PrendoSim additionally considers two, often neglected, physical characteristics: weight distribution and friction, for which appropriate values were selected based on the material that made up each part of the objects. These values are unique for each target object, and they physically match their real-world equivalents as closely as possible; however, a few physics properties were globally set for all objects, *i.e.*, angular and translational drag coefficients, which were set to 0.05, Unity's default setting. Unity provides several collision detection algorithms, from which we selected the "Continuous Dynamics" option, because it optimises the simulation while preventing objects from intersecting one another.

Table 2 presents details about the physical characteristics of each object deployed in the simulator. Static and dynamic friction parameters match the real-world equivalents based on the materials of each part (tool or graspable interface) of the objects. We obtained these coefficients from The Engineering Toolbox[8]. The bounce coefficient was set to 0 for all objects to simulate perfectly plastic contacts, with no rebound speed. Simulating grasps where the object bounces around within the hand after contact is beyond the scope of PrendoSim.

[8]https://www.engineeringtoolbox.com/friction-coefficients-d_778.html

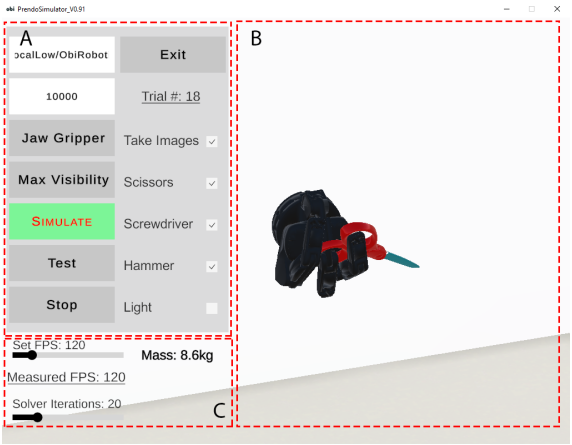

Figure 5: PrendoSim user interface, with A: user input fields, B: 3D visual rendering of the simulation scene, and C: physics engine adjustment fields.

3.4 User Interface

The user interface (UI) is a critical point of our grasp generator. PrendoSim provides an easy-to-use UI, which allows an effortless selection of the various options and parameters for the simulations. The UI has three sections: Section A with major inputs (Fig. 5A); section C with minor inputs (Fig. 5C); and section B with the simulation screen (Fig. 5B). In section A, users are prompted to type the key parameters for the simulation, *i.e.*, number of simulations, gripper type, target object, and the directory path where the data will be stored after the simulation. Each trial is shown in the simulation screen in section B. This section renders the 3D model of both gripper and object, and it shows the current grasp in real time. The user can visually confirm the gripper's configuration and qualitatively evaluate the input parameters. The viewpoint can be regulated via mouse. In section C the user can further adjust simulation-related parameters such as sampling rate and the number of physics solver iterations. Moreover, a display shows the current weight of the object when the algorithm executes point 6 of our strategy, as described in Sec. 3.2.

3.5 Output

PrendoSim outputs the object category, camera pose, and the gripper's pose in a JSON file. We record the condition as the target object involved in the simulation, for example "Screwdriver". We record the

Table 2: Simulated mass (m), static friction parameter (π_s), and kinetic friction parameter (π_k) of the two functional areas (Osiurak et al., 2017) of the three objects in PrendoSim. Friction coefficients μ_s and μ_k are calculated by averaging the friction parameters of the gripper (shown in Table 1) and the object surface it is touching.

	Handle Side			Tool Side		
Object	m (g)	π_s	π_k	m (g)	π_s	π_k
Hammer	70	0.40	0.65	280	0.50	0.30
Scissors	25	0.78	0.25	80	1.05	0.47
Screwdriver	50	1.13	1.40	85	0.35	0.80

camera's view angle onto the scene in world coordinates (position and rotation in quaternions). Then we record the gripper's digit positions and rotation angles in both local (relative to the gripper's root) and world coordinates. The same data is also recorded for the target object. Finally we take note of the target object's categorical orientation, defined in four categories: up, down, pointing away from the hand, and turned with respect to the tool interface.

```
{
  "conditionInfo": "Screwdriver",
  "cameraPose": [posX,posY,posZ,
      rotX,rotY,rotZ,rotW"],
  "jointData": [
  "Digit_1_joint_1,posX,posY,posZ,
      rotX,rotY,rotZ,rotW"
  "Digit_1_joint_2,posX,posY,posZ,
      rotX,rotY,rotZ,rotW"
  "Digit_n_joint_n,posX,posY,posZ,
      rotX,rotY,rotZ,rotW" ]
  "root": [posX,posY,posZ,rotX,rotY
      ,rotZ,rotW"],
  "objectPose": ["screwd(Clone),
      posX,posY,posZ,rotX,rotY,rotZ,
      rotW"],
  "categoricalDir": ["pointing,rotX
      ,rotY,rotZ,rotW"]
}
```

For example in Fig. 6, the tool interface of the screwdriver is pointing away from the gripper, hence it has been labelled accordingly in the output file ("pointing").

Finally we present a series of stable grasps generated by PrendoSim in Fig. 7. We report four grasps per object to show qualitatively the variety of stable grasps generated by PrendoSim. Interestingly, some of the grasps are not human-like (see grasp C1 in Fig. 7) and could not have been generated by following a human-inspired generation process based on learning from demonstration.

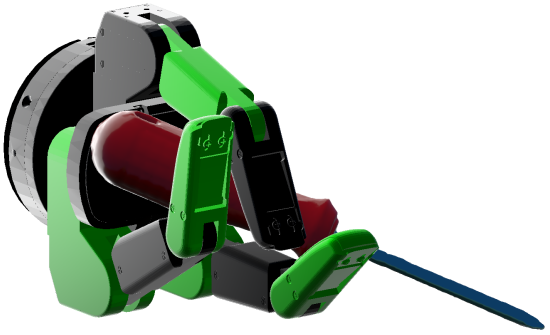

Figure 6: BarrettHand successfully grasping a screwdriver pointing away from the gripper.

4 DISCUSSION AND CONCLUSION

We present a novel grasp generator that accounts for dynamic properties of the gripper and the grasped object, including static and dynamic friction, weight, and weight distribution. PrendoSim is built on Unity's latest physics articulated body component, and it presents two built-in grippers (a parallel jaw gripper and a three-finger hand) and three built-in objects (a hammer, a screwdriver and scissors). PrendoSim utilises the proxy-hand method to compute forces at the contacts and provide an operational definition of grasp stability. The intuitive interface allows the user to easily set up the simulations.

PrendoSim has the great advantage of describing objects realistically, allowing non-uniform materials by defining areas with different dynamic properties. Moreover, it is easy to use, open source and downloadable for free from the following URL (removed for anonymisation). We believe that it can be extended to VR simulation with haptic rendering. In such a case, the kinematic hand would be under the control of the user and thus could be considered a god object, as it is not amenable to exact prediction (Dworkin and Zeltzer, 1993). An algorithm was already proposed where such a user-controlled object retains information about the contact with objects to improve force rendering (Zilles and Salisbury, 1995).

PrendoSim has a few limitations and room for improvement. At the moment, a gripper configuration

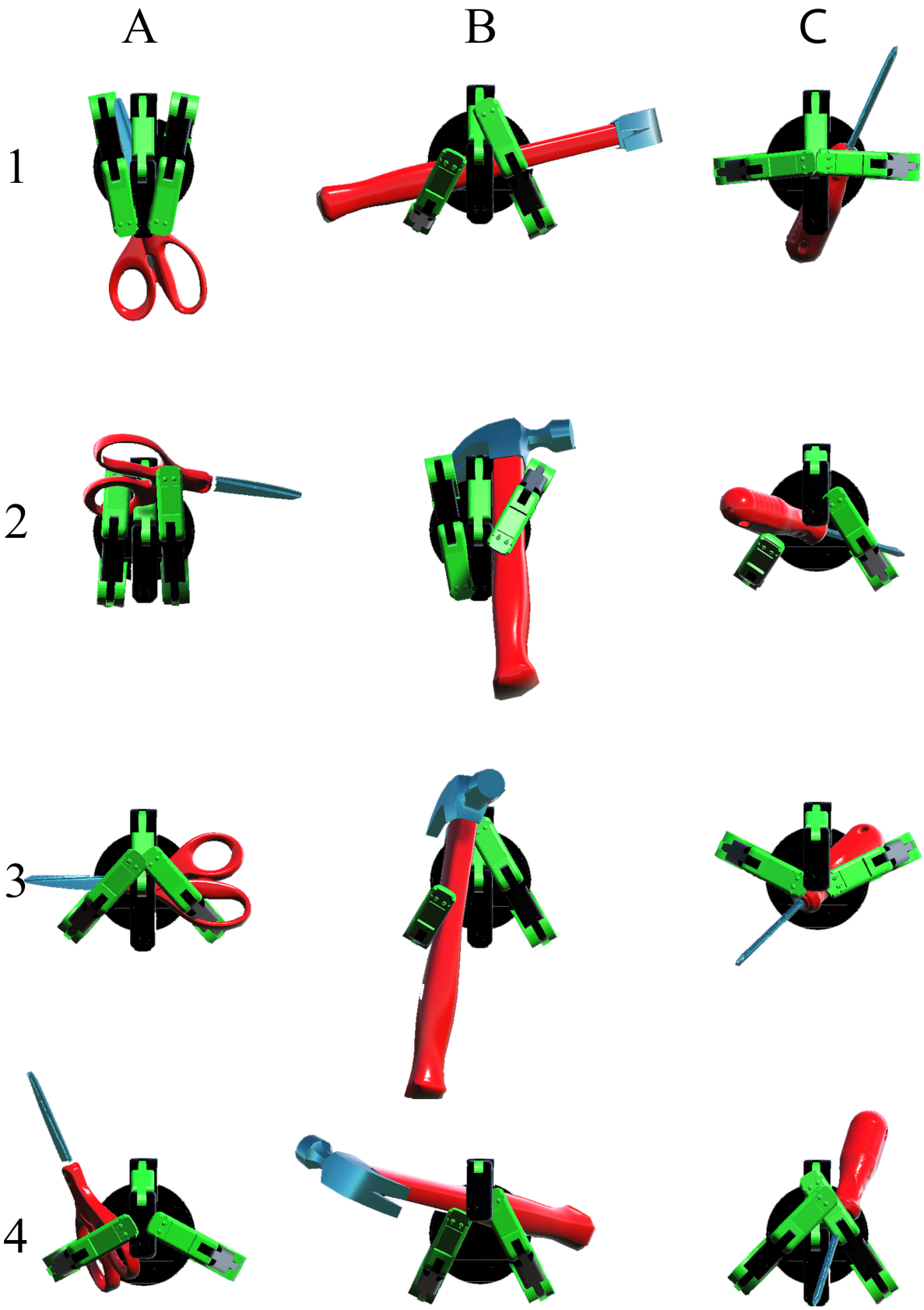

Figure 7: Examples of grasps for each of the target objects arranged in columns A (scissors), B (hammer) and C (screwdriver).

is considered successful if the object remains in the grasp after the hand first closes on it with a moderate grasp force. We are planning to add more grasp stability measures, such as force closure, in order to enable a wider validation of the results. The addition of further objects and grippers would give an even wider range of variation to the simulations. Currently, such additions are possible by directly working on the files in the open-source project of PrendoSim.

To enable more advanced image analysis, such as 3D image reconstruction, our simulator could be extended to include depth-imaging or multi-camera functionalities by adding more cameras in the scene in Unity, which is currently possible with the provided open-source project.

One of the problems with the proxy-hand method, which has been highlighted in haptic rendering, is that the simulated objects are rigid; thus, they have only point contacts. This is a problem especially in human hand simulations, where contact areas are generally wider considering the soft deformations of the fingertips. Contact area simulations can be added to improve realism of contact dynamics, for example to add point torques as in (Talvas et al., 2013).

We believe that PrendoSim is a tool that can benefit multiple communities, as its applications span from the development of grasp algorithms, to the recording of grasp datasets, to the development of VR applications and user study experiments.

REFERENCES

Adjigble, M., Marturi, N., Ortenzi, V., Rajasekaran, V., Corke, P., and Stolkin, R. (2018). Model-free and learning-free grasping by local contact moment matching. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2933–2940, Madrid, Spain.

Ajoudani, A., Zanchettin, A. M., Ivaldi, S., Albu-Schäffer, A., Kosuge, K., and Khatib, O. (2017). Progress and prospects of the human-robot collaboration. *Autonomous Robots*.

Bekiroglu, Y., Marturi, N., Roa, M. A., Adjigble, K. J. M., Pardi, T., Grimm, C., Balasubramanian, R., Hang, K., and Stolkin, R. (2020). Benchmarking protocol for grasp planning algorithms. *IEEE Robotics and Automation Letters*, 5(2):315–322.

Berceanu, C. and Tarnita, D. (2012). Mechanical design and control issues of a dexterous robotic hand. *Advanced Materials Research*, 463-464:1268–1271.

Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: a review. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 348–353.

Borst, C. and Indugula, A. (2005). Realistic virtual grasping. In *Proceedings of the IEEE Conference on Virtual Reality (VR)*, pages 91–320, Bonn, Germany.

Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2017). Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268.

Ding, D., Lee, Y.-H., and Wang, S. (2001). Computation of 3-d form-closure grasps. *IEEE Transactions on Robotics and Automation*, 17(4):515–522.

Dworkin, P. and Zeltzer, D. (1993). A new model for efficient dynamic simulation. In *Proceedings of the Fourth Eurographics Workshop on Animation and Simulation*, pages 135–147.

Ferrari, C. and Canny, J. (1992). Planning optimal grasps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2290–2295.

Gualtieri, M., ten Pas, A., Saenko, K., and Platt, R. (2016). High precision grasp pose detection in dense clutter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605.

Honarpardaz, M., Tarkian, M., Ölvander, J., and Feng, X. (2017). Finger design automation for industrial robot grippers: A review. *Robotics and Autonomous Systems*, 87:104 – 119.

Höll, M., Oberweger, M., Arth, C., and Lepetit, V. (2018). Efficient physics-based implementation for realistic hand-object interaction in virtual reality. In *Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 175–182.

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

Miller, A. T. and Allen, P. K. (2004). GraspIt! a versatile simulator for robotic grasping. *IEEE Robotics Automation Magazine*, 11(4):110–122.

Nguyen, V.-D. (1988). Constructing force- closure grasps. *The International Journal of Robotics Research*, 7(3):3–16.

Nguyenle, T., Verdoja, F., Abu-Dakka, F., and Kyrki, V. (2021). Probabilistic surface friction estimation based on visual and haptic measurements. *IEEE Robotics and Automation Letters*, pages 1–8.

Ortenzi, V., Cini, F., Pardi, T., Marturi, N., Stolkin, R., Corke, P., and Controzzi, M. (2020a). The grasp strategy of a robot passer influences performance and quality of the robot-human object handover. *Frontiers in Robotics and AI*, 7:138.

Ortenzi, V., Controzzi, M., Cini, F., Leitner, J., Bianchi, M., Roa, M. A., and Corke, P. (2019). Robotic manipulation and the role of the task in the metric of success. *Nature Machine Intelligence*, 1(8):340–346.

Ortenzi, V., Cosgun, A., Pardi, T., Chan, W., Croft, E., and Kulic, D. (2020b). Object handovers: a review for robotics. *arXiv*.

Osiurak, F., Rossetti, Y., and Badets, A. (2017). What is an affordance? 40 years later. *Neuroscience & Biobehavioral Reviews*, 77:403–417.

Talvas, A., Marchal, M., and Lécuyer, A. (2013). The god-finger method for improving 3d interaction with virtual objects through simulation of contact area. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, pages 111–114.

Vahrenkamp, N., Kröhnert, M., Ulbrich, S., Asfour, T., Metta, G., Dillmann, R., and Sandini, G. (2012). Simox: A robotics toolbox for simulation, motion and grasp planning. In *International Conference on Intelligent Autonomous Systems (IAS)*, pages 585–594.

Zilles, C. B. and Salisbury, J. K. (1995). A constraint-based god-object method for haptic display. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 146–151.