2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

A Parametric Grasping Methodology for Multi-Manual Interactions in Real-Time Dynamic Simulations

Adnan Munawar, Nishan Srishankar, Loris Fichera & Gregory S. Fischer

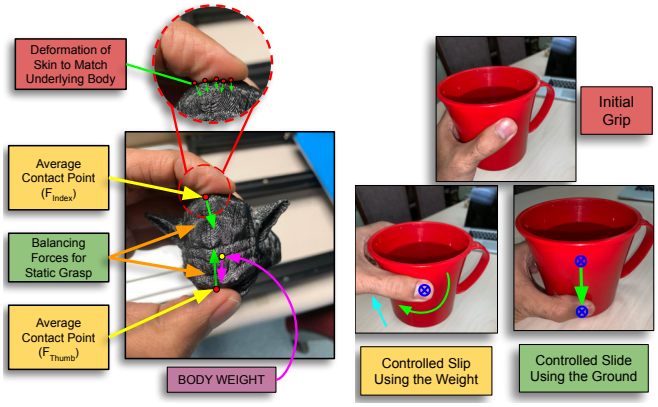
***Abstract*— Interactive simulators are used in several important applications which include the training simulators for teleoperated robotic laparoscopic surgery. While state-of-art simulators are capable of rendering realistic visuals and accurate dynamics, grasping is often implemented using kinematic simplification techniques that prevent truly multi-manual manipulation, which is often an important requirement of the actual task. Realistic grasping and manipulation in simulation is a challenging problem due to the constraints imposed by the implementation of rigid-body dynamics and collision computation techniques in state-of-the-art physics libraries. We present a penalty based parametric approach to achieve multi-manual grasping and manipulation of complex objects at arbitrary postures in a real-time dynamic simulation. This approach is demonstrated by accomplishing multi-manual tasks modeled after realistic scenarios, which include the grasping and manipulation of a two-handed screwdriver task and the manipulation of a deformable thread.**

I. INTRODUCTION

Interactive computer simulations play an important role in robotic teleoperation by enabling human operators to practice and gain experience with a system before operating the actual physical hardware. One important application is training for robotic surgery where new surgeons are trained to get accustomed to the telemanipulated interface. Towards this end, the surgeons can interact with simulated dynamic objects (such as soft-tissues, rigid-body puzzles, peg and hole tasks, etc.) using telemanipulated simulated end-effectors mimicking the actual robot.

An important aspect of natural interaction is the ability to grasp and manipulate dynamic objects. Such interaction includes the dynamic deformation of skin tissue in the vicinity of contact points as shown in Fig. 1(a). To be perceived realistic, interactive simulators need to compute the real-time physics of the interaction between the robot and its environment. This encompasses, among other things, the interplay between the forces created by the user through his/her actions, the forces created by the objects present in the simulated environment, and the resulting changes in the environment.

While most interactive simulators excel at rendering realistic physics, grasping is implemented using simplified techniques that do not mimic natural contact dynamics as shown in Fig. 1(a). As an example, a commonly used technique for simulated grasping [1], [2] deactivates the dynamic properties of the grasped object and treats it as a kinematic body affixed to the grasper. This approach has obvious shortcomings as it limits multi-manual manipulation. Moreover, an important aspect of natural grasping is the ability to allow controlled "slip and slide" of the grasped object as shown in Fig. 1(b). This is useful in training applications, modeled after realistic tasks, that require multi-handed interactions.



(a) (b)

Fig. 1: (a) Natural grasping with skin deformation resulting in increased surface contact. (b) Controlled slip and slide for manipulation by leveraging weight or external obstacles.

We present an approach that utilizes a form of penalty based parametric sensors [3] [4] that can be mounted on any grasper to emulate adequate friction for easy grasping. The concepts presented in this paper were implemented on the Asynchronous Multi-Body Framework (AMBF) [5] using a variety of different simulated graspers and friction surfaces. These graspers were then used to perform complex tasks that involve the manipulation of various dynamic objects, interaction with simulated surgical robots, and controlling deformable bodies (modeled using finite primitives). The implementation is generic and can be expanded to any physics library.

II. RELATED WORK

A summary of notable work addressing the problem of grasping using contact dynamics in simulation is described in this section. GraspIt! [6] by Miller et al. was a prominent simulator that contained multiple robotic hands, contact body dynamics, and a basic grasp planner, but lacked an API

Adnan Munawar is with the Computer Science Department at the Johns Hopkins University, Baltimore, MD, USA. Nishan Srishankar, Loris Fichera & Gregory S. Fischer are with the Department of Robotics Engineering, Worcester Polytechnic Institute, MA, 01609, USA amunawar@jhu.edu, [nsrishankar, lfichera, gfischer]@wpi.edu

This work is supported by the National Science Foundation (NSF) through National Robotics Initiative (NRI) grant: **IIS-1637759** and NSF AccelNet grant: **1927275**

and a modular architecture which limited its functionality. A proposed improvement by Léon et al., called OpenGRASP [7] simulated contact sensors for grasping and attempted to create a realistic simulation of grasping rigid objects using soft contacts.

Moisio et al. [8] used the OpenGRASP toolkit to improve an existing model of simulated tactile sensors that used a soft contact approach without modeling stick-slip. The improvements were made by using a parametric contact force model for surface forces, holding torques and stick-slip as well as generating the sensors using a geometry patch. The resulting simulated framework was compared with real-world robot grasping. However, the authors stated that the computational efficiency of the tactile sensor model could be improved by using a non-brute force collision detection solver. Ciocarlie et al.[9] worked on a general analytical method to model fingertip gripping with friction by analyzing friction constraints on non-planar contacts of elastic materials, formulating a linear complementary problem (LCP), and removing any assumptions about the objects geometries. Goldfeder et al.[10] implemented a grasp planner in GraspIt! that was generalizable to various object/hand geometries/kinematics by decomposing and representing an object as a tree of super-quadratics, hence defining a smaller search space of potentially successful grasps as those which have good grasps on sub-components of the object in the decomposition tree.

Hawkes et al. [11] proposed an alternative to gripping an object purely using the normal force. Their work used gecko-inspired controllable fibrillar adhesives that utilize tangential shear forces mimicking the curvature of an object. Such a gripper could grasp convex objects that are relatively large and featureless as well as delicately grab objects without squeezing. Malvezzi et al. [12] developed a lightweight Matlab toolbox called SynGrasp. In addition to providing an easy way to load hand models, it allowed grasping performance analysis and grasps quality measures such as minimizing contact forces for a given grasp based on a specified cost function.

Interactive training in simulation may involve non-convex shapes and articulated mechanisms that require within-hand and multi-manual manipulation in real-time. By default, the links (belonging to a simulated grasper) have uniform surface friction. As discussed in [13], however, mechanical equivalents of a biological finger can have surface regions with different friction coefficients. We thus present a generalizable approach to interactive training using penalty based contact sensors that complement the underlying collision constraints provided by the physics solver [14].

III. Problem Formulation

For manipulation in real-time dynamic simulations, it is impractical to model grippers such as Fig. 1. Instead, rigid bodies are preferred which require rigid-body collision methods. These methods can be separated based on primitive versus non-primitive collision shapes. Consider a simulation involving spherical bodies that can be modeled using either the corresponding analytical function (primitive shape) or the non-primitive discrete mesh. The collision computation only requires that the center $P^w_{B_i}$ of each **Body B** maintain its radius r_{B_i} from all the other bodies. This constraint can be expressed simply as $||P^w_{B_1} - P^w_{B_2}|| >= (r_{B_1} + r_{B_2})$. To compute the collision forces, the system of equations representing the dynamic simulation is evaluated at time-steps $[n_0, n_1, n_2, ...n_d]$. The resulting error $D_p = r_{B_1} + r_{B_2} - ||P^w_{B_1} - P^w_{B_2}||$, called the penetration depth, is used to calculate a resulting correction force which serves to "repel" the penetrating bodies to re-satisfy the constraint. For a real-time dynamic simulation, the time-step dt is variable, which affects the penetration depth. With this basic setup, the challenges associated with grasping in simulation are as follows:

A. Limitations Associated with Geometric Representation of Rigid Bodies

Simulated rigid bodies have an infinite surface stiffness in the sense that the surface in the vicinity of a contact point for a simulated rigid-body does not deform. Moreover, individual faces (of shapes and meshes representing simulated bodies) are locally smooth whereas the contact surfaces of physical bodies are rough at a microscopic level as illustrated in Fig. 1(a). This roughness plays a major role in both the static and sliding friction response.

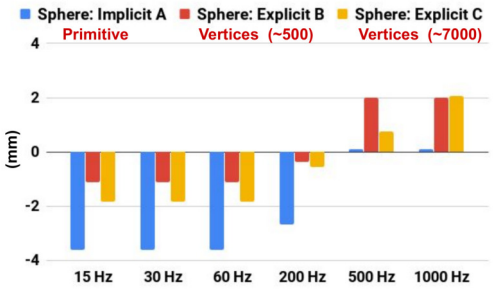


Fig. 2: The penetration depth D_p (maximum fall distance vs equilibrium) of three collision shapes (Spheres) with mass = 50 Kg, radius 0.5 m, non static ground plane position at 0 m and drop height = 2 m.

The inherent softness rendered by collision algorithms, in the form of the penetration depth D_p, in a way counteracts the infinite stiffness of rigid bodies. At a glance, this may be leveraged to mimic the softness inherent to real-world bodies. However, this penetration depth varies based on a few factors such as the length of the time-step dt, the collision shape type and the complexity of the body's geometry for non-primitive shapes. As demonstrated in Fig. 2, a primitive and two non-primitive spherical shapes (of a different number of mesh faces) of identical scale are dropped onto a static plane at various physics update frequencies. The difference in geometry and physics update frequency alters the penetration depth of each object. This varying behavior of the resulting penetration depth D_p limits the use of implicit softness rendered by collision algorithms for contact dynamics purposes.

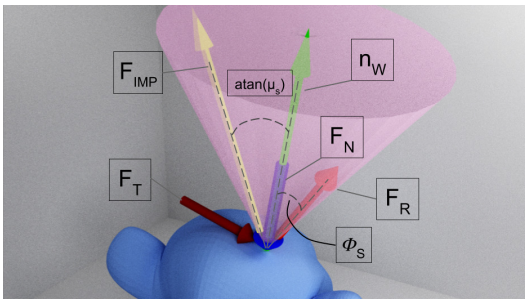


Fig. 3: Visualization of the commonly used friction cone

B. Dynamics Calculation in Physics Libraries

The computation of the normal force F_N is used for the derivation of both the static and sliding friction. The analytical approach to modeling the static friction force, depicted in Fig. 3 is trivial for specifically optimized examples modeled using non-stiff differential equations. On the other hand, for real-time rigid-body dynamics, as in our case, iterative techniques are preferred where the collisions are modeled as instantaneous inequality constraints. Examples of these methods include both velocity (such as Sequential Impulse (SI) [15] [16]) and position based methods (PBD) [17]). These methods counter the penetration by applying corrective impulses (proportional to penetration) or position correction respectively. The application of corrective impulse instead of direct position rectification makes velocity based methods more suitable for simulating friction. However, even for SI solvers, the dependence of the normal force on the penetration depth makes the friction constraint non-linear and not strictly convergent. Moreover, it is difficult to compute the correct contact area for non-primitive shapes without some simplifications. Thus especially for real-time simulations, where the accuracy of the solver has to be comprised to some extent, the varying penetration depth makes even an acceptable friction model using velocity based method insufficient for rendering adequate stick-slip friction.

Penalty based contact modeling approaches [3], which are forms of Force Based methods, are usually avoided in real-time simulations. These methods under-perform in terms of achieving impenetrability as a high enough velocity of a colliding body may result in tunneling [18]. However, in our case, it is their exact penetrability that is leveraged to simplify grasping and emulate stick-slip friction as discussed in the next section.

IV. Methods

The following section provides details about the methods used to model sensor based friction for grasping and manipulation. The section uses several coefficients which are listed in Table I for clarity.

A. Resistive Sensors for Preemptive Contact Computation

Ray Shooting (Ray Tracing) [19], [20] can be used for rendering realistic scenes. Since this technique essentially computes the intersecting point of an object along the path, it can also be used for a variety of other applications as in our case where we combine this ray and some additional parametric data and call it a Resistive sensor (shown in Fig. 4). Each Resistive sensor can be computed independently or in a group which is important as Ray-Tracing is computationally expensive.

TABLE I: Symbols used in this Manuscript.

Symbol	Description
$\vec{F}_S, \vec{F}_V, \vec{F}_N$	Static, Sliding, Normal Friction Force
$\vec{e}_T, \vec{e}_V, \vec{e}_N$	Tangential , Velocity and Penetration Contact Error
μ_S, μ_V	Static and Sliding Friction Coefficient
K_N, K_D	Normal Contact Stiffness and Damping
$\vec{V}^C_{[A,B]}$	General Notation to Express $\vec{V}^C_A$ and $\vec{V}^C_B$ which can be read as $\vec{V}_A$ and $\vec{V}_B$ expressed in C

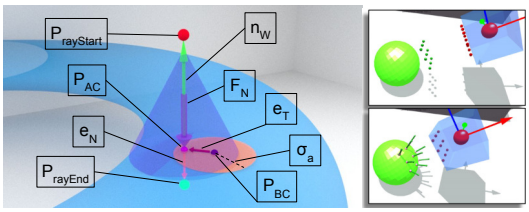


Fig. 4: (a) A single Resistive sensor (left) and the visualization of contact (green spheres) of an array of sensors (red spheres) mounted on a cube.

B. Anatomy of a Resistive Sensor

A Resistive sensor in Fig. 4 visually illustrates some coefficients from Table I. The sensor can be defined w.r.t. a start $\vec{P}_{rayStart}$ and an endpoint $\vec{P}_{rayEnd}$ and is "triggered" when it intersects with another object. In addition to the start point, the sensor is parameterized by a range and an offset (or a depth) from the surface of its parent body. To specify multiple sensors, a separate triangular mesh can be used as discussed later in Alg. 2.

Due to the challenges associated with implementing the static friction cone, as discussed in section III, one alternative is to use a fully deterministic analytical approach represented in Fig. 5. A torque $\vec{\tau}_{[GA,GB]}$ is being applied to both fingers by an external controller. To solve for the contact normal force at points $\vec{P}_{AC}$ and $\vec{P}_{BC}$, we require the knowledge of the torque $\vec{\tau}_{[GA,GB]}$ at J_G, and the lengths $\vec{P}_{GA}$ and $\vec{P}_{GB}$. This example can easily be expanded to non-quasi-static problems, where the gripper is accelerating while holding **Body Z** (Fig. 5). In such cases, the Inertial and Coriolis components need to be considered in addition to the contact dynamics and gravitational components. Furthermore, additional bodies may interact with the gripper's finger and as a result, increase the complexity of contact force computation at $\vec{P}_{AC}$ and $\vec{P}_{BC}$.

Secondly, the vector of Tangential force $\vec{F}_T$ is required to compute the Resultant force $\vec{F}_R$. This tangential force is balanced by the static friction force $\vec{F}_S$ under the constraint $\vec{F}_T <= \vec{F}_S$ and thus preventing tangential displacement. Without using the implicit integration methods, it is not possible to compute the static force before a deflection occurs, which is a challenge to modeling the trivial friction cone. Considering this limitation, σ_a and K_N are utilized.

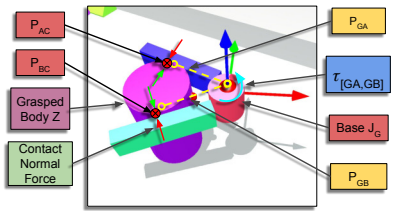


Fig. 5: Grasping an object using a simple two finger gripper.

A displacement along the tangential direction results in error $\vec{e}_T = (T_A^W \vec{Q}_C^A - T_B^W \vec{Q}_C^B)_{proj_T}$, where $proj_T$ is the projection of the contact error in the tangential plane to the direction of the sensor and is then used to compute $\vec{F}_S$ as:

$$\vec{F}_S = \mu_S * \vec{e}_T * \|\vec{F}_N\| \tag{1}$$

Eq. 1 does not rely on the limit $\vec{F}_T <= \vec{F}_S$ but instead on the limiting error $\vec{e}_T$. Once a Resistive sensor triggers as a result of contact with another body, the sensed point $\vec{P}_C^W$ is recorded. This sensed point is then used to calculate $\vec{Q}_C^A$ and $\vec{Q}_C^B$ as:

$$\vec{Q}_C^{[A,B]} = \begin{cases} (T_{[A,B]}^W)^{-1} \vec{P}_C^W, & \text{if } \|\vec{e}_T\| <= \sigma_a, \\ \text{Recompute}, & \text{otherwise.} \end{cases} \tag{2}$$

Here $\vec{Q}_C^A$ is the position of the sensed point $\vec{P}_C^W$ in **Body A** and $\vec{Q}_C^B$ is the sensed point's position in **Body B**. If the limiting condition $\|\vec{e}_T\| <= \sigma_a$ is exceeded, the new value of $\vec{P}_C^W$ is used to compute $\vec{Q}_C^A$ and $\vec{Q}_C^B$ based on Alg. 1.

Algorithm 1 Store Contact Points in Body Frames

```
if Sensor Triggered == True then
    P_C^W = Get Sensed Point()
    e = T_A^W Q_C^A − T_B^W Q_C^B
    if e_{proj_T} > σ_a then
        P_C^[A,B] = (T_[A,B]^W)^{-1} P_C^W
        Q_C^[A,B] = P_C^[A,B]
    end if
end if
```

Similarly $\vec{F}_N$ is calculated as:

$$\vec{F}_N = K_N \vec{e}_N + K_D \delta \vec{e}_N / dt \tag{3}$$

This force is based on the penetration depth of the sensed point w.r.t. the sensor limits. This depth is normalized using Eq. 4 and then projected along the sensor's direction in the World frame.

$$\vec{e}_N = \left(\frac{\vec{P}_C^W - T_A^W \vec{P}_{rayStart}^A}{T_A^W (\vec{P}_{rayStart}^A - \vec{P}_{rayEnd}^A)} \right)_{proj_{N_A}} \tag{4}$$

The terms $\vec{P}_{rayStart}^A$ and $\vec{P}_{rayEnd}^A$ are the start and end points of the Resistive sensor in the **Body A** frame, and $\delta \vec{e}_N / dt$ is to provide damping to the contact normal force. The normal penetration depth for **Body A** is a projection of the contact error onto the normal plane of A ($proj_{N_A}$).

Finally, the sliding friction force is computed from the relative velocities of contact points of **Body A** and **Body B**.

$$\vec{F}_V = \mu_V * \vec{e}_V \tag{5}$$

If the velocity of **Body A** and **Body B** are $(\vec{V}_A^W, \vec{\omega}_A^W)$ and $(\vec{V}_B^W, \vec{\omega}_A^W)$ in the world frame respectively, then:

$$\begin{aligned} (\vec{v}_{C[A,B]}) =& (R_{[A,B]}^W)^{-1} \vec{V}_{[A,B]}^W + \\ & (R_{[A,B]}^W)^{-1} \vec{\omega}_{[A,B]}^W \times T_W^{[A,B]} \vec{P}_C^W \end{aligned} \tag{6}$$

In Eq. 6, the rotation $(R_{[A,B]}^W)^{-1}$ is not collected outside as the cross product $(R_{[A,B]}^W)^{-1} \vec{\omega}_{[A,B]}^W \times T_W^{[A,B]} \vec{P}_C^W$ is not commutative. These velocities are converted back to the World frame using Eq. 7. The $\vec{v}_{C[A,B]}^W$ notation emphasizes that the velocity of sensed point (C) relative to **Body A**, and **Body B**, is expressed in world frame:

$$\vec{v}_{C[A,B]}^W = R_{C[A,B]}^W \vec{v}_{C[A,B]} \tag{7}$$

Which leads to the sliding error:

$$\vec{e}_V = (\vec{v}_{CA}^W - \vec{v}_{CB}^W)_{proj_T} \tag{8}$$

The three components of force are summed together as:

$$\vec{F}_{total} = \vec{F}_N + \vec{F}_V + \vec{F}_S \tag{9}$$

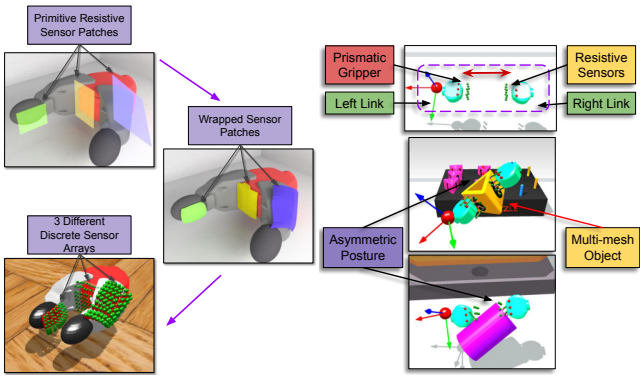
This force can now be used to compute the resulting moment on Body A and Body B by converting it back to Body Frames.

$$\vec{\omega}_{[A,B]}^W = R_{[A,B]}^W ((R_{[A,B]}^W)^{-1} \vec{F}_{total} \times (T_{[A,B]}^W)^{-1} \vec{P}_C^W) \tag{10}$$

The final force and moment are then applied to both **Body A** and **Body B** as action/reaction wrenches. The magnitude of the static friction force calculated from the Resistive sensor can be visualized as an inverse cone shown in Fig. 4(a). Fig. 4(b) shows the deformation of the contact surface formed by Resistive sensors as they penetrate another body. The normal force introduces "softness" to contact dynamics and improves the friction response, which can be leveraged to loosely mimic natural manipulation using soft contacts without the explicit use of deformable body simulations. However, a normal force might not be desirable in certain applications for which Eq. 1 is modified as:

$$\vec{F}_S = \mu_S * \vec{e}_T * \|\vec{e}_N\| \tag{11}$$

Where the depth is the normalized fraction of the sensor's penetration depth. The underlying equations used to represent the behavior of Resistive sensors are based on the combination of both penalty based methods for contact computation and the classical Coulomb friction model. However, these equations are slightly different in their application. This difference results from the fact that instead of two bodies colliding with each other, it is the Resistive sensors mounted on Body A which penetrate Body B. Thus, there is no common normal of collision at this instance, instead, it is the direction of the sensor that is used for the computation of the normal force $\vec{F}_N$ (Eq. 3) as well as the static and sliding friction force $\vec{F}_S$ (Eq. 1) and $\vec{F}_V$ (Eq. 5). The damping K_D stabilizes the normal force by reducing "jitter" that is generally associated with penalty based methods.



(a) (b)

Fig. 6: (a) Skin-wrapping 2D planes for contoured resistive surface. (b) A simple prismatic gripper mounted with Resistive sensors used for grasping at asymmetric postures.

C. Automating Sensor Placement

It is impractical to manually place an array of Resistive sensors on simulated dynamic bodies represented by complex meshes. Thus we propose two approaches for sensor placement, (1) based on the visual mesh of the object and (2) based on a separate parametric mesh specified alongside the visual mesh. These meshes are called the "source meshes" and are used according to Alg. 2. The resulting sensor placement is shown in Fig. 6(a). The primitive patches are wrapped around the visual mesh using Alg. 3.

Based on the vertex and triangle data of the source mesh, the surface is covered using the trivial Alg. 2. This algorithm can be expanded to cover edges and vertices. Using a separate mesh to define the surface of placement has many advantages over using the visual mesh of the object. For gripping tasks, the resistive surface forming the grasp closure is of more interest. Thus a mesh covering only the specific surfaces may be used. Furthermore, the parametric mesh can be defined to smooth out sharp corners of the visual / collision mesh.

Real-world rigid bodies may contain surfaces that are represented by different friction coefficients [13]. While it is possible to create different friction response at different areas on a rigid-body using existing physics APIs, the use of Resistive sensor patches streamlines the process.

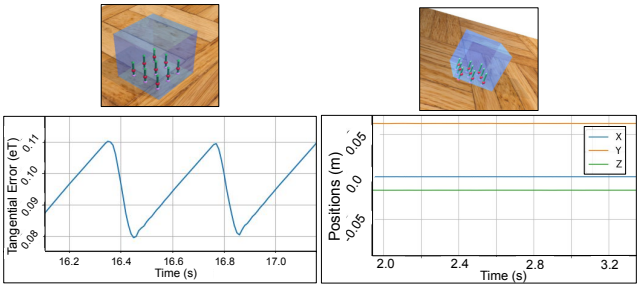
Algorithm 2 Populate Sensors Along Body Surface

$D :=$ Param. Depth, $R :=$ Param. Range, $M :=$ Mesh
$Triangles \leftarrow M$
for $T \in Triangles$ **do**
  $\vec{vtx}_0, \vec{vtx}_1, \vec{vtx}_2 \in T$
  $\vec{edge}_0 = \vec{vtx}_1 - \vec{vtx}_0, \quad \vec{edge}_1 = \vec{vtx}_2 - \vec{vtx}_1$
  $midpoint = \vec{vtx}_0 + \vec{vtx}_1 + \vec{vtx}_2$
  $n_f = \vec{edge}_0 \times \vec{edge}_1 / \|\vec{edge}_0 \times \vec{edge}_1\|$
  $\vec{P}_{rayStart} = midpoint - \vec{n}_f D$
  $\vec{P}_{rayEnd} = \vec{P}_{rayStart} + \vec{n}_f R$
end for

Several free software can be used to generate and modify meshes. Blender is a popular software for graphics design and animation and was used in this study for creating skinned meshes.

Algorithm 3 Wrapping Primitive Around Visual Shape

$S \leftarrow$ Visual Shape, $M \leftarrow$ Primitive Mesh
$\vec{p}_{off} :=$ Param Offset between a body and its sensor mesh
for $v \in M.Vertices$ **do**
  $p_v := v.position$, $n_v := v.normal$
  $\vec{p}_c :=$ Contact Point on S of Ray along n_v
  if Contact Occurred **then**
    $f_c :=$ Nearest Face of S to $\vec{p}_c$
    $\vec{n}_c := f_c.normal$
    if $dot(\vec{n}_c, \vec{n}_v) < 0$ and $\vec{p}_{off} <= (\vec{p}_c - \vec{p}_v)$ **then**
      $\vec{l}_v = \vec{p}_c - \frac{(\vec{p}_c - \vec{p}_v)}{||(\vec{p}_c - \vec{p}_v)||} * \vec{p}_{off}$
      $p_v = l_v$
    else
      Discard v
    end if
  end if
end for
Recompute Normals for M



(a) (b)

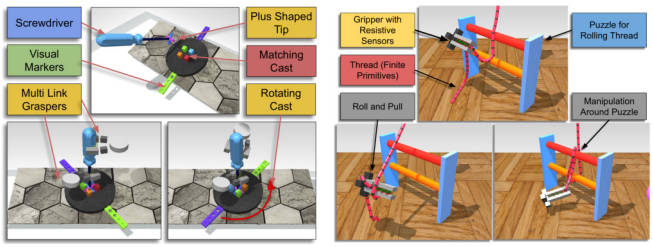
Fig. 7: (a) Constant force stick-slip friction response. (b) Equilibrium on an inclined plane. $m = 0.5Kg$, $K_s = 5000$, $\sigma_a = 0.001$, $K_n = 1$, $K_D = 50$ and $\mu_v = 0.1$.

V. Results

The interactions presented in this section are carried out using various haptic devices and surgical robot controllers [21]. The dynamic environments shown in this section have been designed specifically for the demonstration of manipulation using the proposed methods. The PC setup consists of an Intel(R) Core(TM) i7-3770 CPU (3.40GHz), 32 GB DDR3 RAM (1333 MHz) and an Nvidia GTX 1060 GPU running Ubuntu 18.04. Fig. 7(a) demonstrates the stick-slip phenomenon of a box mounted with Resistive sensors and subject to a constant horizontal force. The stability of the same box is tested by placing it on an inclined plane and then recording its position over time. The response is shown in Fig. 7(b).

The approach for modeling contact dynamics using Resistive sensors presented in this manuscript allows for computing interactions with real-time physics simulations. The interactions are not limited to quasi-static objects but in fact, manipulation can be carried out at high speeds. This is demonstrated with an accompanying video submitted alongside this paper.

Natural manipulation involves grasping complex objects at asymmetric postures and is demonstrated in Fig. 6(b). Such interactions show potential scenarios that are not only applicable to interactive simulators for surgical training but also entertainment and gaming simulators. The procedure of contact dynamics using Resistive sensors takes away the factor of varying geometrical shapes from grasp mechanics.



(a) (b)

Fig. 8: (a) Bi-manual screwdriver manipulation. (b) Deformable thread manipulation.

TABLE II: Parametric Data for Specified Tasks

Task	Obj. Mass (Kg)	μ_s	μ_v	K_N	K_D	$\sigma_a(m)$
Obj. Grasp	0.4-0.8	1000	0.5	1.0	5	0.001
Screwdriver	0.6	5000	0.8	0.8	0.1	0.001
Thread	0.002/Prim.	1000	0.3	0.05	5	0.005

Fig. 8(a) shows a more challenging scenario that mimics a two-handed screwdriver operation. The screwdriver is first grasped and then inserted into the cast (matching the tip shape of the screwdriver) via bi-manual manipulation (controlled via haptic input interface device). After insertion into the cast, the minor hand softens the grip, thereby reducing static friction according to Eq. 1 and the dominant hand rotates the screwdriver to rotate the cast underneath. The task can be carried out repeatedly by tightening the non-dominant hand, softening the dominant hand and re-orienting to a comfortable pose for rotating the screwdriver. Using constraint-based grasping, such a scenario would require pre-planning at the time of picking such that the user holds the screwdriver to accommodate switching between the hands as the dominant hand would require the release of the grasped object by the non-dominant hand for rotation. Lastly, using kinematic simplification by rigidly affixing the screwdriver to either hand would make the two handed rotation impossible.

Similarly, multiple connected objects can be grasped and manipulated. Fig. 8(b) illustrates a task involving the manipulation of a deformable thread around the puzzle. The parametric values used for these three examples are presented in Table. II. The response of the AMBF simulator during the two handed screw-driver operation is shown in Fig. 9. As illustrated, the dynamic frequency of the simulation varies throughout but the Real Time Factor (RTF) stays constant. The Real-Time Factor is an indication that the simulation runs in real-time and can be used for training users since the simulation clock concurrently tracks a real-world clock.

The stiffness achieved through the inclusion of Eq. 3 introduces a soft feel to grasping. This is significant as the objects do not have to be grasped symmetrically as is the case with natural manipulation.

VI. Discussion

We have presented a parametric approach to tackle the problem of grasping and manipulation in simulation using Resistive sensors. The implementation relies on Bullet Physics APIs for appending the calculated friction/normal forces in addition to the internal collision constraints. For the specific demonstrations in this paper, the Sequential Impulse (SI) solver has been used which incorporates the symplectic Euler method for integrating the equations of motions as well the external forces provided by the Resistive sensors. While the initial results are promising, the approach has a few limitations. The first limitation is the number of parameters required to define the friction response of individual Resistive sensors. These parameters vary based on the scope of simulation and require tuning using a combination of empirical and analytical methods. The problem is compounded by the unboundedness of friction coefficients as high values can render the grasp unstable and lower values result in insufficient grip forces.

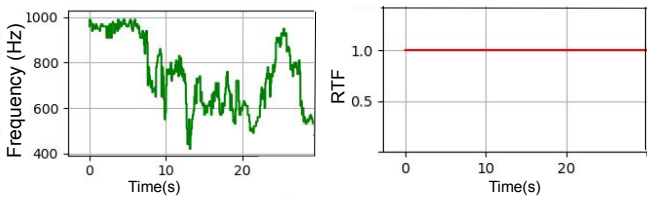


Fig. 9: Real Time Factor for the screw driver task.

Each Resistive sensor requires the calculation of ray-tracing which is an expensive operation. For a large array of Resistive sensors, the required computation time will adversely affect the speed of the dynamics solver for real-time physics. The advent of hardware specialized for Ray Tracing [22] can potentially be leveraged to compute the response from Resistive sensors in parallel. The encapsulation of relevant data for the Resistive sensors means that each sensor can be computed independently at each time-step of the physics simulation. Rather than relying on dedicated GPU hardware, the computation can also be performed in parallel by using multi-threading and batching a group of sensors together.

The next steps involve the implementation of the Resistive sensors for soft-body simulations. We distinguish the term "soft-body" from a "deformable body". Soft-bodies are represented by a single mesh, comprising of connected vertices that form faces. The vertices can interact with other objects in the environment and result in the deformation of the corresponding faces. Deformable bodies, on the other hand, are represented by a finite group of rigid-body meshes (nodes), connected via constraints. Unlike soft-bodies, the deformable bodies have no faces between the inter-connected nodes. We have shown the manipulation of deformable bodies in the paper but more work needs to be done for soft-body simulations. Towards this end, the AMBF Simulator already provides support for soft-body dynamics which should potentially allow a generic implementation for both rigid bodies and soft-bodies.

References

[1] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[2] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, and B. Siciliano, "A v-rep simulator for the da vinci research kit robotic platform," in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pp. 1056–1061, IEEE, 2018.

[3] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1904–1909, IEEE, 2006.

[4] D. C. Ruspini and O. Khatib, "Collision/contact models for dynamic simulation and haptic interaction," in *Robotics Research*, pp. 185–194, Springer, 2000.

[5] A. Munawar, Y. Wang, R. Gondokaryono, and G. S. Fischer, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1875–1882, Nov 2019.

[6] A. Miller, P. Allen, V. Santos, and F. ValeroCuevas, "From robotic hands to human hands: a visualization and simulation engine for grasping research," *Industrial Robot: An International Journal*, vol. 32, no. 1, p. 5563, 2005.

[7] B. Len, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and et al., "OpenGRASP: A Toolkit for Robot Grasping Simulation," *Simulation, Modeling, and Programming for Autonomous Robots Lecture Notes in Computer Science*, p. 109120, 2010.

[8] S. Moisio, B. Len, P. Korkealaakso, and A. Morales, "Model of tactile sensors using soft contacts and its application in robot grasping simulation," *Robotics and Autonomous Systems*, vol. 61, no. 1, p. 112, 2013.

[9] M. Ciocarlie, C. Lackner, and P. Allen, "Soft Finger Model with Adaptive Contact Geometry for Grasping and Manipulation Tasks," *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC07)*, 2007.

[10] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp Planning via Decomposition Trees," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007.

[11] E. W. Hawkes, D. L. Christensen, A. K. Han, H. Jiang, and M. R. Cutkosky, "Grasping without squeezing: Shear adhesion gripper with fibrillar thin film," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[12] M. Malvezzi, G. Gioioso, G. Salvietti, D. Prattichizzo, and A. Bicchi, "Syngrasp: A matlab toolbox for grasp analysis of human and robotic hands," *2013 IEEE International Conference on Robotics and Automation*, 2013.

[13] A. J. Spiers, B. Calli, and A. M. Dollar, "Variable-friction finger surfaces to enable within-hand manipulation via gripping and sliding," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, p. 41164123, 2018.

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

[15] B. Chang and J. E. Colgate, "Real-time impulse-based simulation of rigid body systems for haptic display," in *Proc. Symp. on Interactive 3D Graphics*, pp. 200–209, 1997.

[16] B. V. Mirtich, *Impulse-based dynamic simulation of rigid body systems*. University of California, Berkeley, 1996.

[17] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[18] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," in *ACM Transactions on Graphics (ToG)*, vol. 21, pp. 594–603, ACM, 2002.

[19] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," in *ACM SIGGRAPH computer graphics*, vol. 18, pp. 137–145, ACM, 1984.

[20] Y. M. Govaerts and M. M. Verstraete, "Raytran: A Monte Carlo ray-tracing model to compute light scattering in three-dimensional heterogeneous media," *IEEE Transactions on geoscience and remote sensing*, vol. 36, no. 2, pp. 493–505, 1998.

[21] A. Munawar and G. S. Fischer, "An Asynchronous Multi-Body Simulation Framework for Real-Time Dynamics, Haptics and Learning with Application to Surgical Robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6268–6275, Nov 2019.

[22] Nvidia, "Nvidia ray tracing." `https://developer.nvidia.com/rtx`, 2019. Online.