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

An Open-Source Framework for Rapid Development of Interactive Soft-Body Simulations for Real-Time Training

Adnan Munawar, Nishan Srishankar, and Gregory S. Fischer

Abstract— **We present an open-source framework that provides a low barrier to entry for real-time simulation, visualization, and interactive manipulation of user-specifiable soft-bodies, environments, and robots (using a human-readable front-end interface). The simulated soft-bodies can be interacted by a variety of input interface devices including commercially available haptic devices, game controllers, and the Master Tele-Manipulators (MTMs) of the da Vinci Research Kit (dVRK) with real-time haptic feedback. We propose this framework for carrying out multi-user training, user-studies, and improving the control strategies for manipulation problems. In this paper, we present the associated challenges to the development of such a framework and our proposed solutions. We also demonstrate the performance of this framework with examples of soft-body manipulation and interaction with various input devices.**

I. Introduction

Simulators targeted for Robot-Assisted Surgery are employed for training surgeons to get acclimated to the telemanipulation interface of the robots as well as to enhance the surgical skills. While some applications may focus on the intricate dynamics involved in specific surgical procedures, the majority of training applications involve relatively simpler environments. For example, the peg and hole task [1] is a popular puzzle used in training simulations. The basic building blocks of maneuvering and manipulation required in this task (and similar training tasks), when combined, can be used to generate complex trajectories employed for surgical procedures.

The use of soft-body simulation is of particular interest in surgical training. In this regard, significant research has focused on areas such as dynamics of tissue deformation, rendering visual realism and achieving real-time simulations for specific procedures. However, to the best of our knowledge, surprisingly little work has been done towards the development of a generic framework for integrating custom soft-body simulations with real-time manipulation using input interfaces of surgical robots. Although not initially catering to soft-body dynamics, the Asynchronous Multi-Body Framework (AMBF) [2] was a step in this direction. The AMBF utilized input devices (commercially available haptic devices, gaming controllers and the Master Tele-Manipulators (MTMs) of the da Vinci Research Kit (dVRK) [3]) to interact with user-specifiable dynamic environments. These dynamic environments included not only the training puzzles but also the accurate (both kinematically and dynamically) models of surgical robots. The novel contribution of AMBF over other similar works was the ability to define these environments and robots using a front-end human-readable format [4]. This format, among other things, allowed the easy definition of robots and mechanisms involving parallel linkages, a trait commonly employed in surgical robots.

In this paper, we present an extension to the AMBF simulator (and its specification format) for simulating and interacting with soft-bodies in real-time. We outline the challenges faced in developing such a generic framework and the steps undertaken to solve some of these problems. We also show examples of generating soft-bodies using simple interfaces, reinforcing them internally (for density) and ultimately manipulating them using input interface devices.

II. Problem Formulation

The physical dynamics of deformation and visual realism are important and often focused areas of research for soft-body simulation. This, however, is not the focus of this paper, instead, we discuss some overlooked yet important challenges pertaining to a generic soft-body simulation framework. These challenges include (1) Representation, (2) Visualization, (3) Interaction and Manipulation, and (4) Real-time dynamic update. Representation in this context refers to the description of the soft-body shape and dynamic parameters such as softness, stiffness, bending, and weight distribution. In addition to the geometric shape, the required number of controllable parameters for a simulated soft-body pose a challenge for description purposes. Simulators such as Gazebo [5] and V-REP [6] utilize physics computation libraries that support soft-body simulation, however, they do not support soft-body representations or their visualizations. Visualization is more challenging as compared to representation. Unlike rigid bodies that do not require a constant update to the mesh geometry, soft-body meshes require computationally expensive updates at each simulation step. Thus the algorithm for updating the visual representation of the soft-body adds additional overhead to the simulation step. High-density meshes are preferred for visual realism, however, they are problematic for soft-body simulation. For this reason, a pair of meshes may be utilized: (1) a high-quality mesh for visualization and (2) a lower resolution mesh to represent the soft-body. The meshes can then be fused together such that the lower resolution mesh can

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

be used to update the vertices of the high-quality mesh. Finally, the interactions and manipulation of the soft-body are complex tasks that are usually specialized for specific soft-bodies. Realistic grasping is a challenging problem in rigid body dynamics and holds true even for soft-body simulations.

III. Related Work

While the implementations of soft-body simulations in surgical/real-time scenarios are limited, we referenced papers that looked at other applications such as gaming. Maciel et al. [7] used the NVIDIA's PhysX library to provide bimanual interactivity in simulated surgical operations with implicit integration-based physics processing (between 10 to 20 Hz). However, at the time of the paper (2009), the PhysX source code was closed and only recently became open-source. Danevičius et al. [8] worked on the gamification of a soft-body simulator that simulated soft-body physics (based on particles interconnected with constant-length, flexible springs) and performed computational offloading on a cloud to obtain high-quality graphics (rendering in frames-per-second, and number of particles representing a body) in real-time, however, lacked collision modeling and was limited to simplistic soft objects. Tan et al.[9] manipulated soft-body characters using Finite-Element methods to simulate the deformation and muscle fibers to control the shape/motion. To move these objects, the authors created controllers to obtain an objective function that was passed to an optimization solver. The optimization solver, in turn, determined new muscle fiber lengths and whether points of contact should be static or sliding. This resulted in a combination of muscle contractions that enabled the soft-body to perform various locomotions. Mesit et. al [10], [11] modeled soft bodies with pressure based forces and proposed using implicit integration methods and externally limiting the forces to solve failure cases when large constraints in stiffness or time were applied.

IV. Methods

The Asynchronous Multi-Body Framework offers a real-time dynamic simulation that can be manipulated with multiple input interface devices. It utilizes external libraries such as Computer Haptics and Interface (CHAI-3D) [12] and Bullet Physics [13]. This paper extends the prior rigid body simulation framework to incorporate the specification, simulation, and manipulation of soft-bodies. The inclusion of the soft-body support to existing specification interfaces of AMBF is shown in Fig. 1. In the rest of this section, we discuss the methodology used to address the problems described in Section II.

A. Real-Time Simulation of Soft-Body Dynamics

Soft-bodies can be represented using a collection of nodes with inertial properties interconnected directly or indirectly to other nodes. Each node is subject to the laws of dynamics and can collide with other objects in the environment. The interconnections between two nodes can be generalized with a constraint formed by a three-dimensional spring which

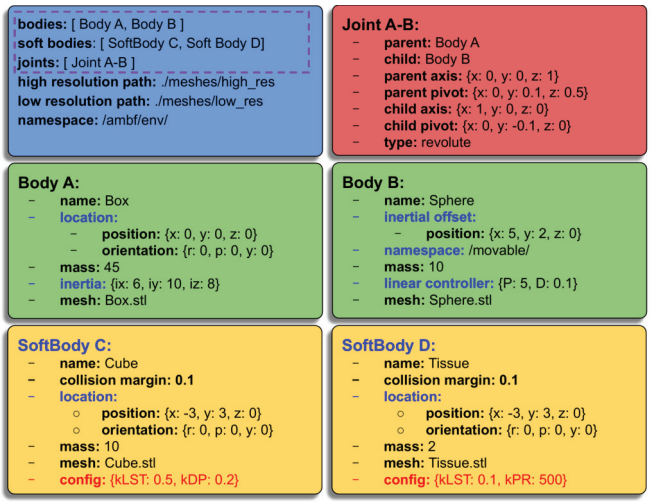


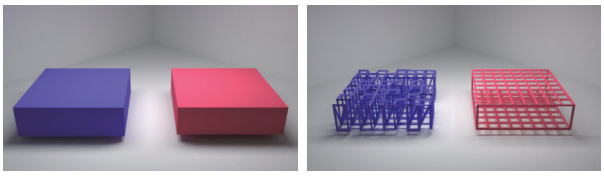
Fig. 1: Anatomy of the AMBF format. The blue tile forms the header and consists of global parameters and header lists which are highlighted with the purple dotted border. The red tile represents a constraint, green represents rigid bodies and yellow represents soft-bodies. The tunable parameters for soft-body dynamics can be set using the config parameter highlighted in red. The defined parameters include $kLST$ = Linear Stiffness Coefficient, kDP = Node Damping Coefficient, kPR = Internal Pressure Coefficient. The location is defined in global coordinates, while inertial offsets, pivots and axis are in local (body) coordinates.

consists of tension, torsion, and flexion. The combination of constraints due to motion, collision, and contact dynamics can be modeled using different methods which can be categorized into direct force computation, velocity-based methods (Sequential Impulse Constraints [14], position-based methods (PBD) [15], and indirect representation as Linear Complimentary Problems (LCPs) [16]. The position of each node is updated at every step of the dynamics simulation based on the explicit or symplectic Euler method (implicit method is used less often). The time-step dt between each update is an important factor affecting the accuracy of the solution. In real-time dynamic simulations, where collision computation is a factor of the number of bodies in contact, it is impractical to fix the step-time dt to a preset value.

Mixed soft-body and rigid-body simulations with real-time dynamic updates pose challenges to implementation as stability and convergence are not guaranteed. In our previous work [2], we simulated rigid-body dynamics with the implicit variation of time step dt to keep in sync with the real-world (Wall) clock and explicitly varying the number of sub-iterations N according to Equation 1:

$$dt < \delta t_i \times N \quad ; \quad N \in Z^+ \quad \& \quad N \leq N_{max} \qquad (1)$$

We employ a similar method in this paper, while maintaining bounds on the number of sub-iterations $N < N_{max}$ and the actual integration sub-step size $\delta t_i := 1/(Hz_{fixed})$. N_{max} is tunable parameter and is set to 10.



(a) (b)

Fig. 2: Two meshes with similar surface geometry but different internal structure defined using the OBJ mesh format.

B. Representation of a Soft-Body

As discussed in Section IV-A, soft-bodies can be represented by inertial nodes that are interconnected to other nodes. Meshes used in computer simulations also employ a similar form of interconnection of vertices defining the surface. However, meshes for soft-body representation (defined by nodes) also comprise of an internal lattice forming the skeletal structure of the soft-body. An example is shown in Fig. 2 showing two similar bodies with equivalent subdivisions along the surface, however one body (on the left) has an internal skeletal structure while the other (on the right) does not.

Rather than developing a new specification for a mesh, we have utilized existing standards of mesh representations for defining the basic geometry of a soft-body. The following mesh formats, along with their advantages and disadvantages for soft-body representation, have been utilized.

- **The StereoLithography (STL) Mesh**: STL is an older and widely used format. It supports the definition of triangular faces by specifying three vertices as floating point numbers. A normal is specified for each triangle. Since each triangle defines its vertices explicitly, the shared vertices are repeated for each triangle.
- **The Autodesk (3DS) Mesh**: 3DS is proprietary and a binary format. Similar to STL, 3DS defines the mesh geometry as triangles containing a maximum of three vertices. 3DS can also define scene objects such as camera, lighting and textures.
- **The Wavefront (OBJ) Mesh**: The OBJ format has more features compared to STL and 3DS representations and is a non-proprietary format. The biggest advantage for soft-body representation is that each face can comprise of more than 3 vertices. Similarly, edges without faces can be defined as polylines. All the vertices are specified as separate lists, and therefore, vertex repetition can be avoided although is not mandated in the format. The format also supports face normals and per-vertex texture data.

For representation purposes, all three formats have been supported. However, Wavefront's OBJ is the preferred mesh format since it has features to define both the surface as well as the internal structures using polylines. For visual realism, a pair consisting of a visual (high-density) mesh and a low resolution collision mesh is used. As shown in the AMBF format in Fig. 1, soft-body properties are specified alongside the mesh definition.

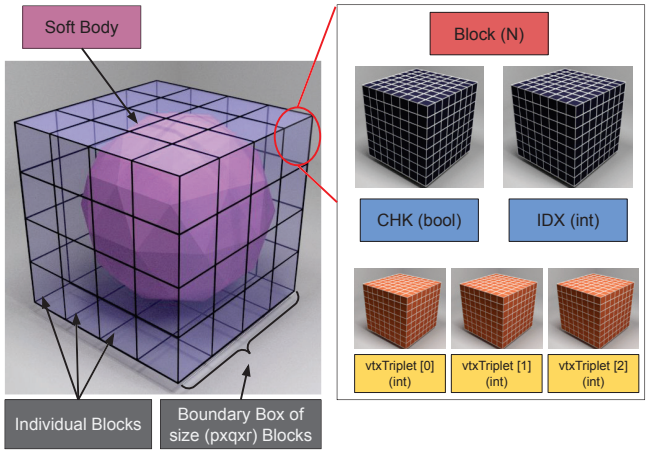


Fig. 3: Reference image for Alg. 1. The soft-body fits in the boundary box that is sub-divided into p, q and r blocks along x,y and z axes respectively. Each block in then parsed individually by creating 5 sub-blocks (CHK, IDX and 3 Vertex Triplet sub-blocks).

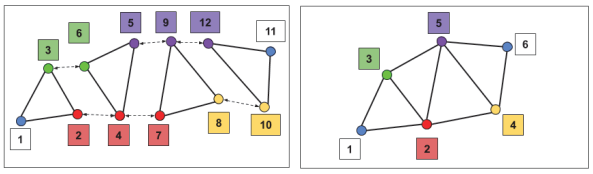


Fig. 4: (a) The original vertex indices that do not account for repeated vertices. (b) The reduced vertex list with the duplicate vertices unified together into a new list.

C. Visualization

As discussed in Section IV-B, a pair comprising of a visual and a collision mesh is used to represent the soft-body. The collision mesh can be defined by using any of the three supported mesh formats. The array of vertices retrieved using these formats may include repeated vertices that need to be unified for proper soft-body representation. The brute force approach to counting repeated vertices is computationally exponential and undesirable. Instead, hashing techniques are employed that turn the vertex unification into an almost linear problem. These techniques are more or less modification of the "vertex weld" [17] algorithm in which the spatial vertices are discretized into smaller bins (sub-blocks). The resulting welded mesh has a reduced vertex count. An extra step is required to modify the triangle indices forming the faces of the mesh to map ("rewire") to the reduced set of vertices. This extra step is not directly related to the vertex welding algorithm.

The unification of repeated vertices is necessary for soft-body dynamics but such a reduced mesh might not be desirable for a generic rendering application (which uses the original mesh with repeated vertices for visualization). Therefore, a different approach, which is in part based on vertex welding, is proposed which unifies the repeated vertices and stores them in a data structure. This algorithm

Algorithm 1 Vertex Triplet Generation

s_b = No. of Vertices Per Block
$n_b = n_{vtx}/s_b$ ▷ n_{vtx} = No. of Vertices
$r_{[x,y,z]} = n_{vtx}/dim_{[x,y,z]}$ ▷ $dim_{[x,y,z]}$ = x, y, z Dim.
$vC[n_{vtx}] \leftarrow False$ ▷ vC = Vtx Checked
$vT[n_{vtx}][3] \leftarrow -1$ ▷ vT = Vtx Triplet
$CHK[s_b][s_b][s_b] \leftarrow False$
$IDX[s_b][s_b][s_b] \leftarrow -1$
for b_x = 0 to n_b **do**
  for b_y = 0 to n_b **do**
    for b_z = 0 to n_b **do**
      $l_{[x,y,z]} = b_{[x,y,z]} \times s_b$
      $h_{[x,y,z]} = l_{[x,y,z]} + s_b$
      $CHK[s_b][s_b][s_b] \leftarrow False$
      $IDX[s_b][s_b][s_b] \leftarrow -1$
      for i = 0 to n_{vtx} **do**
        if $vC[i] == False$ **then**
          $p \leftarrow vtx.position$ ▷ $k_{x,y,z}$ = Key
          $k_{[x,y,z]} = r_{[x,y,z]} * (p - min_{[x,y,z]})$
          if $k_{[x,y,z]} \in [l_{[x,y,z]}, h_{[x,y,z]}]$ **then**
            $k_{[x,y,z]} = k_{[x,y,z]} - dim_{[x,y,z]}$
            $vC[i] = True$
            $vT[i][0] = i$
            if $CHK[k_x][k_y][k_z] == False$ **then**
              $CHK[k_x][k_y][k_z] = True$
              $IDX[k_x][k_y][k_z] = i$
              $vT[i][1] = i$
            else
              $vT[i][1] = IDX[k_x][k_y][k_z]$
            end if
          end if
        end if
      end for
    end for
  end for
end for

also stores the relation between unified vertices and their original non-unified copies.

The algorithm discussed above can be divided into two sub-algorithms which include (1) the Vertex Triplet Generation (Alg. 1) and (2) Generating Unique Triangle Indices (Alg. 2). The first algorithm fills a data structure consisting of three arrays (3 dimensional sub-blocks). This data-structure is called the **Vertex Triplets** and its three associated arrays are described as follows:

- **Original Vertex Indices** $vtxTriplet[0]$ The first array contains the indices to original vertices that form the mesh.
- **Unified Vertex Indices** $vtxTriplet[1]$ The second array contains the vertex indices referring to the first index at which the vertex occurred in the original vertex list.
- **New Vertex Indices** $vtxTriplet[2]$ Finally the third array contains indices from a newly formed array containing only the distinct vertices.

TABLE I: Population of Vertex Triplets

vT[0]	1	2	3	4	5	6	7	8	9	10	11	12
vT[1]	1	2	3	2	5	3	2	8	5	8	11	5
vT[2]	1	2	3	2	4	3	2	5	4	5	6	4

Fig. 3 visualizes the data structures mentioned in Alg. 1. The **Vertex Triplets** for the mesh triangles shown in Fig. 4 are presented in Table. I. Afterward, Alg. 2 is used to compute rewired triangle indices corresponding to the newly sorted vertices and edges. The Alg. 1 is used instead of external "vertex welding" libraries due to the requirement of preserving the interconnection between the duplicate vertices on the high-resolution visual mesh with the unified vertices on the lower-resolution collision mesh. Both the algorithms have been tested on several meshes with known unique and duplicate vertex counts and work as expected. That being said, there may be areas where the algorithms can be optimized.

Algorithm 2 Generating Unique Triangle Indices

$C_u = 0$, $I_b = 0$, $I_c = 0$ ▷ C_u = Unique Vtx Counter
$vtx_{tree} = [n_{uvtx}][]$ ▷ n_{uvtx} = No. Unique Vertices
for i = 0 to n_{vtx} **do**
  if $vT[i][1] == vT[i][0]$ and $vT[i][2] == -1$ **then**
    $vT[i][2] = C_u$ ▷ vT = Vertex Triplet
    $vtx_{tree}[C_u] \leftarrow i$
    C_u++
  else if $vT[i][1] < vT[i][0]$ **then**
    $I_b = vT[i][1]$
    $I_c = vT[I_b][2]$
    $vT[i][2] = I_c$
    $vtx_{tree}[I_c] \leftarrow i$
  else if $vT[i][1] > vT[i][0]$ **then**
    $I_b = vT[i][1]$
    if $vT[I_b][2] == -1$ **then**
      C_u++
      $vT[I_b][2] = C_u$
    end if
    $vT[i][2] = vT[I_b][2]$
  end if
end for

D. Manipulation of Soft-Body

Soft-body grasping and manipulation involve a sequence of steps. To develop a generic implementation to grasp any soft-body, we have used simulated sensors (based on ray-tracing elements) that are placed on the simulated graspers to detect proximity to collision objects. The ray-tracing algorithm is used in computer simulations to trace out the path of light rays as they repeatedly collide with objects in the simulation. Become of the geometric implementation, ray-tracing can be used to detect the nearest points between two surfaces. In a trivial ray-tracing implementation, the starting point of the ray originates at the light source, which

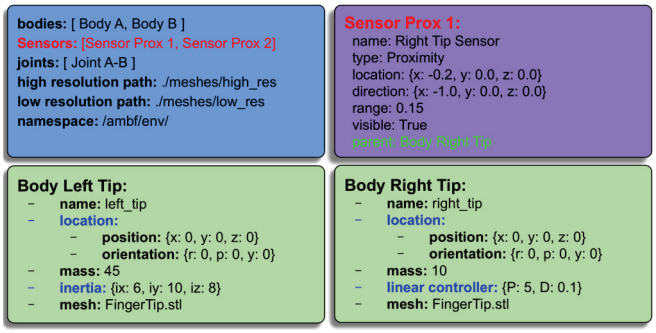


Fig. 5: Proximity sensors can be defined using the same method as bodies, joints, and scene objects in Fig. 1. The proximity sensor is parented to the desired body with the relative location offset, direction, and range.

is usually fixed. However, for the nearest point calculation in dynamic objects, the rays can be parented to a dynamic body. Therefore each proximity sensor has the attributes shown in Fig. 5.

If a proximity sensor "triggers" and the grasping closure angle is less than a user-specifiable threshold, the contact face in the ray's path is computed. Then, the nodes forming the face are anchored using Alg. 3. An anchor is a simulated constraint that attaches the vertices forming the nearest soft-body face to the parent body on which the sensor is mounted. To prevent jerk on newly anchored vertices, the offset between the parent body and the vertices is stored and then used as the desired offset until the anchor remains intact. This anchor produces a weak force according to Alg. 4 that guides the connected vertices along the parent body.

Algorithm 3 Anchor Vertices to Parent

G := Sensor's Parent
Face ∈ Get Nearest Face to Contact Point
Vertices ∈ Face
$T_G^W \in$ G.Transform
for v ∈ Vertices **do**
  $P_v^W \in$ v.Pos
  $P_v^G = (T_G^W)^{-1} P_v^W$
  a ← Anchor(v, P_v^G)
  G.Anchors.append(a)
end for

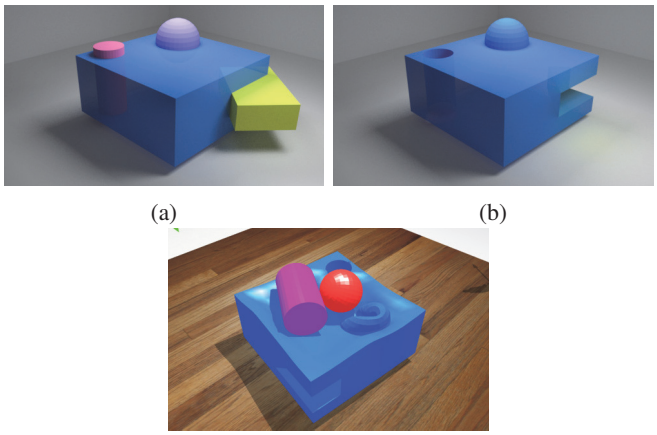
V. Results

As discussed in Section I, soft-body simulation for surgical training is a multi-fold problem, four of which have been identified in this paper. We discuss the results relating to these four challenges in the following section.

Many free software can be used for creating meshes that represent the geometry of soft-bodies. We have selected Blender [18] which already has an existing plugin for generating rigid bodies, robots, and scenes for the AMBF Simulator [19]. Blender is also supported across all major operating systems and offers a highly customizable interface.

Algorithm 4 Update Anchored Vertices

for G ∈ Rigid Bodies **do**
  $T_G^W \in$ G.Transform
  H_a, D_a := Param. Anchor Hardness and Damping
  for a ∈ G.Anchors **do**
    $P_v^{G'} \in$ a.Offset
    $v \in$ a.Vertex
    $P_v^W \in$ a.Offset
    $P_v^G = (T_G^W)^{-1} P_v^W$
    $\delta P_n = P_v^{G'} - P_v^G$
    $F_a = H_a \delta P_n + D_a \frac{\delta P_n - \delta P_{n-1}}{dt}$
    a.ApplyForce(F_a)
  end for
end for



(a) (b)

(c)

Fig. 6: Similar to convex hulls used for rigid body dynamics, a complex soft-body shape can be generated using a compound of simpler shapes. These simple shapes can be fused together using mesh boolean (add or subtract) operations (a) → (b). Finally, (c) shows the simulation and interaction of this mesh in AMBF.

We begin by either creating primitives or importing existing meshes in Blender. These meshes can be sub-divided, morphed, and cut using the existing tools provided in Blender. The detailed discussion on mesh manipulation and features in Blender is beyond the scope of this paper and many tutorials are available for this purpose. Fig. 6 (a) and (b) shows the operations of converting a primitive mesh into a compound shape using simple Boolean operations. Similarly, Fig. 7 shows the conversion of simple cylindrical shape into a body resembling a meatloaf with textured surface.

It is challenging to impart volumetric constraints to soft-body meshes and thus depending on the simulation context different approached may be used. These approaches include 1) constraints based on the volume of a convex mesh [20] and constraint based on the estimated pressure [21]. Another approach is to model a skeletal structure/lattice inside the original mesh. Tetgen [22] is a useful library which allows such mesh skeletalizations. However, for this discussion, we

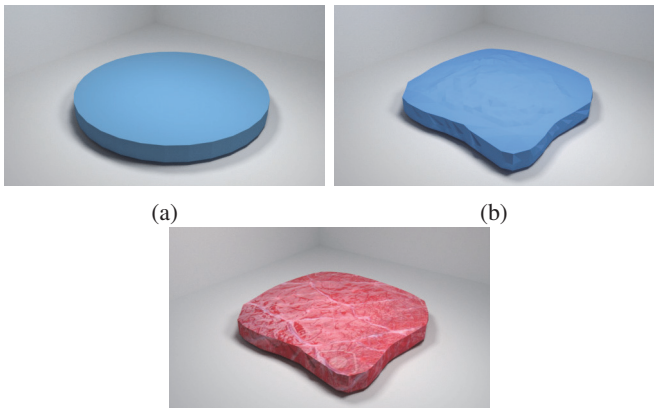


Fig. 7: Sequential process of converting a cylindrical primitive to a mesh with coarse surface, creating internal edges for structural stability and finally applying texture for visual realism.

explicitly model the internal structure of the mesh by connecting the desired vertices. Vertex interconnection interior to the mesh surface only requires the creation of edges and not faces. As discussed in Section IV-B, most mesh representation formats only support faces and not edges explicitly. Fig. 2 shows the original visual mesh with an interconnected lattice which was created using generic tools in Blender.

The front-end specification format described in SectionIV-B conveniently separates the soft-body geometry from its properties. These properties specify attributes such as flexion, torsion and elongation and specified in the config field displayed in red in Fig. 1. The description file can be loaded in the AMBF simulator to generically support soft bodies that are manipulable with multiple input devices in a real-time simulation.

We show the manipulation of various soft-bodies with teleoperated grippers controlled by dVRK MTMs in Fig. 8. The methodology used for grasping is not modeled based on natural interaction as it does not rely on friction for pinching and manipulation. That being said, grasping soft-bodies using the methodology presented in Section IV-D provides a simplistic, yet useful, approach to interactive manipulation. It should be noted that this methodology leverages the closest pinched faces of soft-bodies as described in Alg. 3. The adhering force can be scaled by changing the value of H_a & D_a in Alg. 4. Fig. 9 illustrates the Real Time Factor (RTF) during the example of soft-cloth manipulation shown in Fig. 8.

VI. Discussion

In this paper we have shown the design methodology of a framework targeted for real-time simulation and manipulation of user-specified soft-bodies using existing mesh representation formats. While this framework can potentially provide a convenient research platform for the research

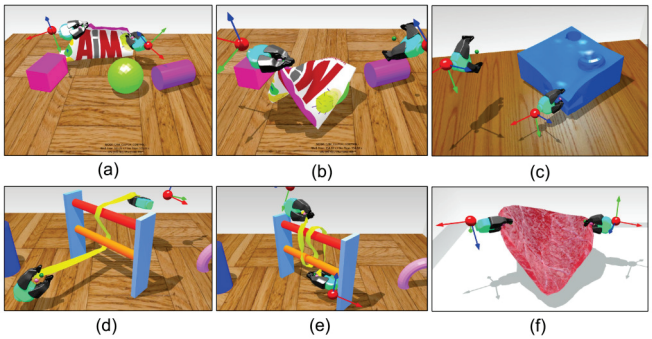


Fig. 8: Examples of Soft-body manipulation using the dVRK MTMs

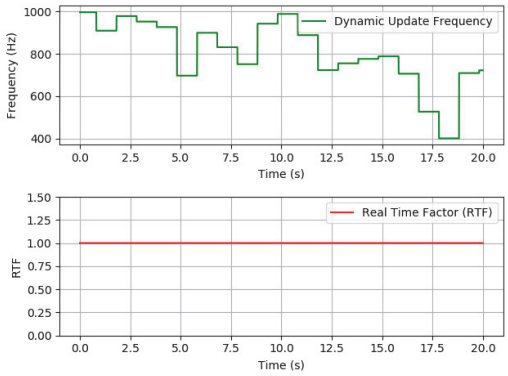


Fig. 9: Real Time Factor for tasks shown in Fig. 8 (a), (b).

community working towards surgical robotics, several additional features need to be added. These missing features include collision detection for faces in addition to vertices, specification of heterogeneous properties, simulating visco-elastic friction, emulating breathing meshes (i.e. intermittent compression/decompression via dynamic volumetric / pressure constraints) and simpler interface to fixing some soft-body nodes in the world.

Other necessary features catering to surgical robotics include the support for cutting, stitching and shearing. We propose the extension of AMBF's sensor interface to include cutting sensors. Similar to proximity sensors (Fig. 5), cutting sensors can potentially be mounted to a rigid-body and can be used the sub-divide and dissect the connecting links at the contact point. This is also a future goal. Stitching is a more challenging problem to address using generic methods that can be implemented at a framework level and therefore needs further exploration.

Lastly, there is a need for developing soft-body environments that can replicate actual surgical sub-tasks while using our proposed framework. Although this task mostly entails the speciality required to design such environments rather than the validity of the proposed framework, it is imperative in demonstrating the actual use-case of the AMBF soft-body framework. In that sense, we consider this is as an immediate future goal.

REFERENCES

[1] A. Ghasemloonia, Y. Maddahi, K. Zareinia, S. Lama, J. C. Dort, and G. R. Sutherland, "Surgical skill assessment using motion quality and smoothness," Journal of surgical education, vol. 74, no. 2, pp. 295–305, 2017.

[2] 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.

[3] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci surgical system," in IEEE Intl. Conf. on Robotics and Auto. (ICRA), (Hong Kong, China), pp. 6434–6439, 2014.

[4] 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.

[5] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3, pp. 2149–2154, IEEE, 2004.

[6] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in Proc. of The International Conference on Intelligent Robots and Systems (IROS), 2013.

[7] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De, "Using the physx engine for physics-based virtual surgery with force feedback," The International Journal of Medical Robotics and Computer Assisted Surgery, vol. 5, no. 3, p. 341353, 2009.

[8] E. Daneviius, R. Maskelinas, R. Damaeviius, D. Poap, and M. Woniak, "A soft body physics simulator with computational offloading to the cloud," Information, vol. 9, p. 318, Nov 2018.

[9] J. Tan, G. Turk, and C. K. Liu, "Soft body locomotion," ACM Transactions on Graphics, vol. 31, p. 111, Jan 2012.

[10] J. Mesit and R. K. Guha, "Simulation of soft bodies with pressure force and the implicit method," First Asia International Conference on Modelling and Simulation (AMS07), 2007.

[11] J. Mesit. PhD thesis, Department of Electrical Engineering and Computer Science in the College of Engineering and Computer Science at the University of Central Florida, 2010.

[12] F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, J. Warren, O. Khatib, and K. Salisbury, "The chai libraries," in Proceedings of Eurohaptics 2003, (Dublin, Ireland), pp. 496–500, 2003.

[13] E. Coumans, "Bullet physics simulation," in ACM SIGGRAPH 2015 Courses, SIGGRAPH '15, (New York, NY, USA), ACM, 2015.

[14] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in Proceedings of the 1995 symposium on Interactive 3D graphics, pp. 181–ff, ACM, 1995.

[15] 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.

[16] R. Tonge, L. Zhang, and D. Sequeira, "Method and program solving lcps for rigid body dynamics," July 18 2006. US Patent 7,079,145.

[17] L. Velho, "A dynamic adaptive mesh library based on stellar operators," Journal of Graphics Tools, vol. 9, no. 2, pp. 21–47, 2004.

[18] B. O. Community, Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[19] A. Munawar, "AMBF." https://github.com/WPI-AIM/ambf, Sep 2019. Github.

[20] Y. Duan, W. Huang, H. Chang, W. Chen, J. Zhou, S. K. Teo, Y. Su, C. K. Chui, and S. Chang, "Volume preserved mass–spring model with novel constraints for soft tissue deformation," IEEE journal of biomedical and health informatics, vol. 20, no. 1, pp. 268–280, 2014.

[21] M. Matyka and M. Ollila, "Pressure model of soft body simulation," in The Annual SIGRAD Conference. Special Theme-Real-Time Simulations. Conference Proceedings from SIGRAD2003, no. 010, pp. 29–33, Linköping University Electronic Press, 2003.

[22] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," ACM Trans. Math. Softw., vol. 41, pp. 11:1–11:36, Feb. 2015.