An Integrated Design Pipeline for Tactile Sensing Robotic Manipulators

Lara Zlokapa¹, Yiyue Luo¹, Jie Xu¹, Michael Foshey¹, Kui Wu², Pulkit Agrawal¹, and Wojciech Matusik¹

http://robohands.csail.mit.edu/

Fig. 1. Our Pipeline: A manipulator is (A) generated using grammar rules and (B) interactively re-shaped via cage-based deformation. Next, the (C) user specifies touch sensor placement. The sensors and hand components are (E) knitted and (D) 3D printed prior to (F) manual assembly of the manipulator.

Abstract—Traditional robotic manipulator design methods require extensive, time-consuming, and manual trial and error to produce a viable design. During this process, engineers often spend their time redesigning or reshaping components as they discover better topologies for the robotic manipulator. Tactile sensors, while useful, often complicate the design due to their bulky form factor. We propose an integrated design pipeline to streamline the design and manufacturing of robotic manipulators with knitted, glove-like tactile sensors. The proposed pipeline allows a designer to assemble a collection of modular, open-source components by applying predefined graph grammar rules. The end result is an intuitive design paradigm that allows the creation of new virtual designs of manipulators in a matter of minutes. Our framework allows the designer to fine-tune the manipulator’s shape through cage-based geometry deformation. Finally, the designer can select surfaces for adding tactile sensing. Once the manipulator design is finished, the program will automatically generate 3D printing and knitting files for manufacturing. We demonstrate the utility of this pipeline by creating four custom manipulators tested on real-world tasks: screwing in a wing screw, sorting water bottles, picking up an egg, and cutting paper with scissors.

I. INTRODUCTION

Currently designing robotic manipulators with tactile sensing is a time-consuming and manual process. One typically brainstorms designs to solve a specific task (or set of tasks), prototypes a selection of the brainstormed designs, chooses the most promising prototype(s), and repetitively iterates prototyping until a successful design is achieved. The majority of time is spent on design iterations driven by trial and error: several attempts may be necessary before a functional prototype is produced. These iterations often require both mechanical modifications (involving designing and testing new parts from scratch) and altering the topology of existing parts upon the realization that a part with different shape or size may be better suited for the task. Topology changes are time consuming because re-shaped pieces may no longer fit together physically, or component mates and parametric constraints in the CAD program may break after dimensions are modified. When this happens, a human must re-model components, identify and fix parametric constraints, and manually re-connect CAD assembly pieces to re-assemble the manipulator model.

Integrating tactile sensors in a robotic manipulator further complicates the already human-labor intensive design process. Many tactile sensors [1], [2] are bulky and cannot be simply added on top of existing designs. Instead, manipulators must be designed around them. This adds further geometric constraints on component interface sizing to the already tedious process of modifying part geometries using traditional CAD programs. From start to finish, depending on task and design complexity, it may take months or years to produce a high-quality robotic manipulator.

We propose a pipeline with an interactive user interface to streamline the design and manufacturing process which is illustrated in Fig. [1]. Our pipeline enables the user to design task-specific, cable-driven robotic manipulators with pressure sensing in a matter of minutes. Using our open-source collection of modular sub-component 3D models and the proposed grammar rules for assembly, users can quickly create many different robotic manipulator CAD models. Because the connections between sub-components are encoded in the grammar rules, a complete 3D model of the manipulator updates in real-time as the user applies grammar

1Authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, 32 Vassar St, Cambridge, MA 02139, USA laraz@mit.edu
2Author is with Lightspeed & Quantum Studios, Tencent America, 12777 Jefferson Blvd, Building E, Los Angeles, CA 90066, USA
rules. No manual assembly of components in a CAD program is required. Then, using an intuitive, cage-based deformation method, the user may topologically deform (i.e., lengthen, widen, and otherwise distort) the manipulator models according to the desired finger and hand shape and size. If needed, at this stage, users can identify small regions for placing tactile sensors on the manipulator’s surface. The sensors are built into a knitted cover that conforms around the manipulator like a glove.

Our design pipeline guarantees that each manipulator design can be manufactured. Once the manipulator is virtually completed, our program automatically generates manufacturing files for 3D printing manipulator components and files for automatically manufacturing the tactile glove via an industrial knitting machine. By removing many manual and time-consuming steps in the traditional approach (e.g., modeling CAD components, laboriously assembling components in CAD, adapting CAD models during geometry changes, and tediously integrating touch sensors), our proposed pipeline enables designers to focus on improving form rather than fixing functionality.

We evaluated the efficacy of our design pipeline on four tasks chosen to demonstrate the breadth of possible designs and the ability to integrate tactile sensors. We designed and manufactured four separate manipulators to (1) pick up an egg, (2) screw on a wing screw, (3) sort water bottles, and (4) cut paper with scissors.

II. RELATED WORK

Manipulator Design. Existing robotic hands such as the Shadow Hand, DLR Hand [3], UW Hand [4], RBO Hand 2 [5], and others [6], [7] have been designed using traditional methods for a specific set of tasks. Changing the end task would require complete redesign which is likely to consume months if not years. For instance, large joints may not be manufacturable at a small scale, or task-specialized manipulator fingers may not be cross-applicable for other tasks, requiring new brainstorming, testing, and specialized design.

Modular robot design is an effective way to generate various robot structures from a small library of base components. It has been applied to generate whole robots ([8], [9], [10]) and modular hands, including ModGrasp [11], OpenMRH [12], and NSU’s sensorized, pneumatic robotic hand [13]. These hands, though modular, generally rely on a single standard finger model that tessellates to extend or shorten the finger. Most similar to ours is the Yale OpenHand that offers a library of components for assembly [14], [15]. However, previous works have a limited component selection, resulting in a limited set of possible topologies. Additionally, they only explore the discrete topology space of robot designs. In contrast, our system considers both the discrete topology of the manipulator and the continuous geometry of each component thus providing a richer design space. Finally, none of these modular hands employ grammar or allow for custom deformation.

Grammar-based design paradigm has previously been employed to generate simulated multi-pedal robots [16], mathematically model the self-assembly of robotic systems [17], create IKEA cabinets and tables [18], and generate passive dynamic brachiating robots [19]. However, grammar driven design not been employed for creating manipulators nor has it been tested in the real-world. To the best of our knowledge, our work consists of the first demonstration of an integrated computational framework for the design of robotic manipulators and sensor placement.

Tactile Sensing. Many commonly available tactile sensors have form-factor and compliance restrictions that impact the geometry and structure of manipulator design. The Robonaut 2 Hand has rigid sensors built into the palmar side of the hand’s phalanges [20], [21]. The iCub Hand is conformally covered with flexible PCB acting like a capacitive pressure sensor at the finger tips [22]. The RBO 2 Hand is wrapped in liquid metal strain sensors to calculate the deformation and extrapolate contact with the grasped objects [23]. Finally, some hands may be fitted with BioTac sensors [24], ready-made sensorized fingertips that are fitted on the robotic hand in place of the original hand’s finger tips. These sensors are available in a single size and their form factor cannot be altered if a different manipulator topology is required. Other common tactile sensors are the Tekscan Grip system [25], Gelsight and other vision-based silicone sensors [1], [26], [27], [28], and the biomimetic multimodal sensor [29], [30]. However, none of these can both be designed and manufactured in a computer-automated manner and conformally cover the robot hand of complex geometry. Electronic skins [31], [32], while more flexible and adaptable, have not been scaled up to larger sizes due to the delicate manual manufacturing processes.

In this work, we incorporate computational design and digital fabrication of knitted pressure sensing matrices to conformally cover our manipulators in a scalable, cost-efficient manner. We hope this will enable broader exploration of tactile sensing for object manipulation.

III. DESIGN WORKFLOW

Our design workflow is summarized in Fig. 1. We designed a library of components (Fig. 2) that can be combined using the proposed context-sensitive grammar (Sec. III-A) to create a diverse array of manipulators. From this discrete design space, a manipulator topology is chosen. Next, the component shapes can be refined (Sec. III-B) to increase manipulator’s suitability for the desired task. The design requirements of grammar drive composition, ease of shape deformation, and the guarantee on manufacturing impose constraints on component designs that are discussed in Sec. III-C. Finally, Sec. III-D details how the user can specify the location of touch sensors.

A. Context-Sensitive Grammar

We represent a manipulator assembly design as a graph where each node corresponds to a physical sub-component and each edge encodes a connection between two sub-components (e.g., relative rotation, translation, etc.). This choice of graph representation guarantees that each assembly
has a unique graph, and each graph corresponds to a unique assembly. The task of generating diverse manipulator designs therefore reduces to generating diverse graphs.

Our manipulator grammar consists of two sub-grammars, a palm grammar and a finger grammar, with rules defined in Fig. 3. The palm grammar generates palms of different sizes, shapes, and numbers of finger slots. Once the palm grammar produces a palm, the user proceeds with the finger grammar to add fingers from the palm. Each grammar consists of:

1) **Terminal symbols** (noted as lowercase letters). These represent the nodes and edges of a graph.

2) **Non-terminal symbols** (noted as uppercase letters). These represent sub-assemblies or sections of a graph.

3) A **start symbol**. A non-terminal symbol that initializes the design.

4) **Expansion rules**. These convert non-terminal symbols into other non-terminal and terminal symbols. They allow the creation of many different graphs based the order and selection of rules applied.

The terminal and non-terminal components used to create the manipulators are shown in Fig. 2 with their associated letter symbols. Note that we only show the components required to make the manipulators in this paper; the “library” of components can be augmented as desired.

Palm Grammar: To design diverse palms, we compose components shown on the left side of Fig. 2. These components can be connected in a planar grid using the palm grammar rules to generate palms of varying shapes (see Fig. 4). It should be noted that each palm grammar rule can be applied in three configurations rotated by $90\degree (l \in 1, 2, 3)$ degrees to expand the palm in different directions. For example, a knuckle node (k) can be connected to either left, right, top and bottom to a connector node (C). Once the palm has been built, it serves as a start symbol for the finger grammar to attach fingers if desired.

Finger Grammar: Similar to the structure of human fingers, the finger grammar combines finger components depicted in Fig. 2 linearly: components are added distally to the fingers to “grow” them in length until termination with a fingertip.

Fig. 4 shows a few grammar-generated manipulator designs. With only thirteen components, our grammar can produce myriad manipulator configurations with different palm or finger shapes. On the order of $10^{10}$ unique fingers can be generated from fifteen finger expansion rules and six terminal finger components, assuming the fingers are constrained to lengths of at most three segments. Restricting the palm to a three by three grid (only for calculation purposes) with up to six fingers, at least $10^{49}$ unique hands exist within our constraints. Such a broad design space can be drastically increased with additional grammar pieces. To efficiently explore the vast design space, we developed an interactive design interface (Fig. 1(A)).

B. Deformation-Based Geometry Shape Design

Once the hand’s topology is established, our pipeline proceeds to the shape refinement phase, where the user may change the geometry of individual components. While the grammar generates discrete manipulator topologies, the specific dimensions of the manipulators may be sub-optimal for the desired task. For instance, it may be beneficial for the phalanges to be longer or for the fingers to taper. Our geometric deformation method supplements the discrete grammar-based designs by enabling users to quickly and intuitively vary the manipulator’s shape in a continuous manner to further optimize their design. To allow users to easily make
design deformations that span multiple shape dimensions, its logical to use a low-dimensional design parameterization.

In traditional CAD modeling, users would manually parametrize each feature’s dimensions, a mistake prone process. Inspired by Xu et al. [33], we apply cage-based deformation to parameterize the manipulator. This technique encloses each high-resolution subcomponent mesh into a cage-like coarse and cuboid shape mesh. The shape of the enclosed subcomponent can be altered by simply moving the cage vertices to scale, shear, and taper while guaranteeing that connections with other surrounding subcomponents are preserved, thereby ensuring manufacturability. We adapt Xu et al.’s method to build an intuitive user interface (UI) for interactively modifying the shape of the manipulator as shown in Fig. 1(B). The UI allows users to manipulate the cage mesh vertices of each subcomponent, deforming the manipulator’s underlying high-resolution mesh.

C. Grammar Component Design

Each grammar component is associated with three meshes. First, there is a high-resolution mesh used for 3D printing and manipulator renderings. Second, a coarse, cuboid cage mesh encloses each component and is used to specify deformation. Third, to generate the knitted sensors, there is another low-resolution, cuboid mesh that approximates the associated high-resolution mesh shape. Both the high resolution and knitting meshes are deformed by changes to the cage mesh. Appendix II contains more details on these meshes.

To preserve the mechanical relationships between parts during deformation, components correspond to mechanical systems (e.g., static phalanges or dynamic joints) rather than to physical parts (e.g., phalanges with joints) as shown in Fig. 6. In the figure, two phalanges that are attached by a joint are separated into three grammar components (Fig. 6, middle): two phalanx shafts and one pin joint. The joint consists of three physical pieces: the distal end of one phalanx, a pin, and the proximal end of another phalanx. The joint component can only be scaled uniformly or axially during cage-based deformation to ensure it functions as a joint. Shearing or scaling in any other axis will result in elliptical pins, which cannot pivot and therefore do not work, breaking the mechanical relationship between the components in the joint. In contrast, the shaft (phalanx) of a finger may be stretched and sheared in almost any manner without impeding its function. Since deformation cages enclose and control each component, dividing functional aspects into separate components with different deformation considerations allows users to maximally deform the assembly without compromising functionality. This choice of part division, combined with careful grammar rule selection, ensures that any design can be manufactured.

D. Tactile Sensing Cover Design

Given the embedded cuboid mesh from the deformation stage, we offer a design interface based on the stitch meshes framework [34], [35] for users to place sensors and generate corresponding knitting files (Fig. 1(C)). To ensure the manipulator can wear the knitted cover, users specify the sides of the finger that the sensor wraps around. From this, our system generates a planar knitting pattern with a quad dominant stitch mesh based on the pattern’s edge length. Each stitch face in this quad-dominant stitch mesh [34] represents a real stitch in the knitted cover: a quad represents a knit and a pentagon represent an increase or decrease stitch to make the knitted structure conform to the component mesh. Users select a stitch face to place a sensor. Finally, our system automatically generates machine knitting files: it traces a knitting path and specifies stitches based on if the face is a knit, increase/decrease, or a sensor.

Our knitted sensors were manufactured based on [36]. Each knitted cover requires two layers, one with horizontally and the other with vertically integrated piezoresistive fibers. By overlapping the two layers, we form a sensing matrix: sensing points are located at the intersections of the orthogonal overlapped fibers where the piezoresistive nanocomposite is sandwiched by two conductive electrodes. The tactile sensors convert pressure stimuli into electric signals, which a customized read-out circuit acquires.

IV. FABRICATION

The grammar subcomponents were designed in SolidWorks then imported into the proposed program. As shown in the accompanying video, in the program, we manually applied grammar rules to create manipulator graphs, deformed the manipulators to the desired shape, and specified all tactile sensing points. The program generated STL and DAT files for 3D printing the hand and knitting the sensors. After manufacturing, the printed pieces were assembled, cables strung, and sensors stitched closed over the manipulators. Each manipulator was mounted on a motor box of Dynamixel motors, which was then mounted on a UR5 arm.

A. Hand Structure

The program-generated STL files were 3D printed on a Markforged printer using Onyx, a micro-carbon fiber filled nylon, and assembled after printing. Bushings were added in the palm using holes in the Connector subcomponents to guide the cables. Spectra cables were threaded from each motor, through the manipulator, to the finger joints, and back to the motors that they originated from, creating a fully-actuated, closed-loop cable drive. After threading and tying all the cables, the system was tensioned by sliding each motor down a short track on its mount until the cables on the motor became sufficiently tight (approx. 20-30 N). The motor was then tightened to prevent it from sliding.
A sensorized four-finger hand picks up an egg from a table, shakes it to demonstrate the secure grasp, and places it back on the table.

1) Task description: A sensorized four-finger hand picks up an egg from a table, shakes it to demonstrate the secure grasp, and places it back on the table.

2) Design: This design was selected based on engineering intuition as the most secure method of holding an egg. The manipulator fingers are mounted on a forked finger with angled joints to conform around the egg when they bend. The two lower fingers take advantage of our continuous deformation: they are significantly wider at the tip to better cradle the wider base of the egg. The deformed width of the manipulators was determined by measuring the egg, and the design required only one attempt. Six sensors are located on the inner surface of each of the four fingers that contacts the egg.

3) Control: From a set location, the hand closes its fingers around the egg until the sensors on most of the fingers reach an experimentally-determined pressure threshold. It follows a hard-coded sequence to shake and release the egg.

4) Performance: The manipulator picks up the egg and grasps it securely so that the egg does not move when the robotic arm shakes it. It releases the egg without any breakage. We note that grasp is reliable enough that we never broke or dropped any eggs during testing. A sample grasping sequence with a tactile sensing signal is shown in Fig. 8.

B. Screwing on a Wing Screw

1) Task description: In our demo, a single, sensorized, rigid finger mounted on a UR5 arm screws a wing screw into a hole until tight. Requiring no moving parts (other than the UR5 wrist), this is one of the simplest designs that the design space can produce.

2) Design: Consisting only of a single, rigid finger, the manipulator is designed to minimize the number of moving parts (i.e., only rotation of the UR5 wrist) while ensuring that it is possible to screw in the wing screw. The base of the finger is lengthened and offset (sheared, using our cage deformation) to prevent the finger from colliding with the nut while the fingertip contacts the wing. The width of the finger accommodates several wing screw sizes. Sensors are located on the contact surface between the wings of the wing screw and the finger.

This design required only two iterations. The first iteration was a straight finger, which required more complicated control of the UR5 arm. The second iteration (seen in Fig.
fingers were tapered using cage deformation to prevent the "expand" to fill the finger hole, stabilizing the scissors. The office scissors, the "pointer" and "middle" fingers abduct to attempting to cut the material, and it will stop cutting. Tactile sensors detect that excessive force is required when between the scissor blades instead of a sheet of paper, the scissors then cuts with them. If a hard material is placed been completed.

Sensor feedback determines when each cut has
dons most shapes of office scissors) and dexterity in handling

D. Cutting Paper with Scissors

This scissor manipulator demonstrates grip adaptability (it
don most shapes of office scissors) and dexterity in handling the scissors. Sensor feedback determines when each cut has been completed.

1) Task description: A three-finger manipulator first dons scissors then cuts with them. If a hard material is placed between the scissor blades instead of a sheet of paper, the tactile sensors detect that excessive force is required when attempting to cut the material, and it will stop cutting.

2) Design: Since scissors are created for human hands, a design with a thumb and two opposing fingers was selected. To accommodate finger holes of varying shapes and sizes in office scissors, the “pointer” and “middle” fingers abduct to “expand” to fill the finger hole, stabilizing the scissors. The fingers were tapered using cage deformation to prevent the scissors from slipping towards the base of the palm. Flexed distal joints ensure that the scissors do not slip off the fingers.

This design required the most iterations to cut with any brand of office scissors. Three configurations were tested, including two types of non-articulated fingers for the larger scissor handle hole. The configuration seen in this paper required two iterations, where the fingertip was narrowed to accommodate narrower scissor handle holes.

3) Control: To don the scissors, the robot fully extends the fingers. The scissors are placed on the robot hand, then the robot abducts the two fingers in the larger scissor handle hole and flexes all distal joints for a secure grasp. To cut, the robot opens the scissors by spreading its thumb, moves the hand forward a prescribed amount, and flexes the thumb to close the scissors until the pressure threshold is reached.

4) Performance: The manipulator is able to don a variety of office scissors and cut paper with them, stopping cutting when the paper has been cut. It detects when a hard surface is put between the scissor blades and does not cut it. The accompanying video demonstrates the scissors cutting through paper and rejecting cutting hard acrylic sheets.

VI. DISCUSSION AND CONCLUSION

In this paper, we presented a design pipeline for creating a variety of robotic manipulators and demonstrated application of the design method with four manipulators for four tasks. A user interface enabled the user to design the manipulator’s morphology using a context-sensitive grammar, topologically deform it, and specify tactile sensing points. The program then automatically generates files for manufacturing, and the user assembles the manipulator with the knitted sensors. This self-contained method simplifies the manipulator design process by providing a grammar that allows the user to flexibly arrange and re-arrange components in a speedy manner while ensuring that all component configurations result in manufacturable designs. Additionally, it shortens the redesign time by providing a basis of pre-tested components, giving the user confidence that initial designs will perform.

While the pipeline performed well for both design and manufacturing, we hope to improve the sensors and motor box. Though functional and easy to manufacture, the sensors require substantial manual tuning in every repetition of each of the four tasks. Improving the accuracy and reliability of sensors is an important area of future investigation. We also found that the manipulator cables were subject to breakage at high loads. In principle, this problem can be resolved by increasing the cable diameter and improving the attachment method so that the cable does not pass over any sharp 90° bends at the motor.

Our design pipeline has applications beyond manual design. The computer-friendly graph and grammar representation enables interfacing with ML, AI, and other optimization and simulation software. For instance, it may be integrated with the already-developed AI-driven geometry and control optimization [33] to quantitatively optimize topology and control for user-specified manipulator configurations. Alternately, it may be integrated with an algorithm that
methodically searches and simulates the design space to determine the optimal manipulator. The ease with which this program can be integrated with other computational processes opens many opportunities for co-optimization on this program can be integrated with other computational methods to determine the optimal manipulator. The ease with which a program can be methodically searched and simulated in the design space to be designed for interaction, as shown in "The International Journal of Robotics Research, vol. 31, no. 13, pp. 1531–1555, 2012.

Z. Xu, X. Kumar, and E. Todorov, "The 2d voxel: A low-cost, 2d voxel-driven hand with fast and compliant actuation," The International Journal of Robotics Research, 2013.

R. Deimel and O. Brock, "A novel type of compliant and undercuttable robotic hand for dexterous grasping," The International Journal of Robotics Research, vol. 35, no. 1-3, pp. 161–185, 2016.

D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," Nature, vol. 521, no. 7553, p. 467, 2015.

C. Piazza, G. Grioli, M. Catalano, and A. Bicchi, "A century of robotic hand designs for ease of fabrication and adoption," Autonomous Robots, vol. 37:1–37:12, 2012.

J. Xu, X. Chen, H. Xue, C. Zhang, D. Guo, and H. Liu, "A dual-modal vision-based tactile sensor for robotic hand grasping," in 2015 IEEE/RSJ International Conference on Robotics and Automation (ICRA), 2018, pp. 4740–4745.

N. Wettels and G. E. Loeb, "Haptic feature extraction from a biomimetic tactile sensor: Force, contact location and curvature," in 2011 IEEE International Conference on Robotics and Biomimetics, Dec 2011, pp. 2471–2478.

P. Park, M. Kim, Y. Lee, H. S. Lee, and H. Ko, "Fingertip skin–inspired microstructured ferroelectric skins discriminate static/dynamic pressure and temperature stimuli," Science advances, vol. 1, no. 9, p. e1500661, 2015.

A. Chortos, J. Liu, and Z. Bao, "Pursuing prosthetic electronic skin." Nature materials, vol. 15 9, pp. 937–50, 2016.

M. C. Boutry, M. Negre, M. Jorda, O. Vardoulis, A. Chortos, O. Khatib, and Z. Bao, “A hierarchically patterned, bioinspired e-skin able to detect the direction of applied pressure for robotics,” Science Robotics, vol. 3, no. 24, 2018.

J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, “An End-to-End Differentiable Framework for Contact-Aware Robot Design,” in Proceedings of Robotics: Science and Systems, Virtual, July 2021.

C. Yuksel, J. M. Kaldor, D. L. James, and S. Marschner, “Stitching meshes for modeling knitting with yarn-level detail,” ACM Trans. Graph. (Proceedings of SIGGRAPH 2012), vol. 31, no. 3, pp. 37:1–37:12, 2012.

V. Narayanan, K. Wu, C. Yuksel, and J. McCann, “Visual knitting machine programming,” ACM Transactions on Graphics (TOG), vol. 38, no. 4, pp. 1–13, 2019.

Y. Luo, Y. Li, P. Sharma, W. Shou, K. Wu, M. Foshey, B. Li, T. Palacios, A. Torralba, and W. Matusik, “Learning human–environment interactions using conformal tactile textiles,” Nature Electronics, vol. 4, no. 3, pp. 193–201, 2021.

E. Klavins, R. Ghrist, and D. Lipsky, “Graph grammars for self-assembly of robotic systems,” in IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04, 2004, vol. 5. IEEE, 2004, pp. 5293–5300.

M. Lau, A. Ohgawara, J. Miitan, and T. Igarashi, “Converting 3d furniture models to fabricatable parts and connectors,” ACM Transactions on Graphics (TOG), vol. 30, no. 4, pp. 1–6, 2011.

F. Stöckl and K. Shea, “Automated Synthesis of Passive Dynamic Brachiating Robots Using a Simulation-Driven Graph Grammar Method,” Journal of Mechanical Design, vol. 139, no. 9, 071023, 2017. [Online]. Available: https://doi.org/10.1115/1.4037255

L. B. Bridgewater, C. A. Ihke, M. A. Diffiter, M. E. Abdallah, N. A. Radford, J. M. Rogers, S. Yawathi, R. S. Askew, and D. M. Linn, “The robonaut 2 hand - designed to do work with tools,” in 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 3425–3430.

C. A. Ihke, M. A. Diffiter, D. M. Linn, R. Platt, and B. K. Griffith, "Phalanx tactile load cell," U.S. Patent 7 784 363 B2, 2010.

N. Jamali, M. Maggioli, F. Giovannini, G. Metta, and L. Natale, “A new design of a fingertip for the iubd hand,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 2705–2710.

V. Wall and O. Brock, “Multi-task sensorization of soft actuators using prior knowledge,” in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 9416–9421.

J. A. Fishel and G. E. Loeb, “Sensing tactile microvibrations with the biotic — comparison with human sensitivity,” in 2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), 2012, pp. 1122–1127.

A. Chortos, J. Liu, and Z. Bao, “Pursuing prosthetic electronic skin.” Nature materials, vol. 15 9, pp. 937–50, 2016.

M. C. Boutry, M. Negre, M. Jorda, O. Vardoulis, A. Chortos, O. Khatib, and Z. Bao, “A hierarchically patterned, bioinspired e-skin able to detect the direction of applied pressure for robotics.” Science Robotics, vol. 3, no. 24, 2018.
APPENDIX I
MANIPULATOR GENERATION

In this section, the grammar rules used for each manipulator are referred to by number only where Rp refers to palm grammar rules and Rf refers to finger grammar rules. The rule associated with each rule number can be found listed in Fig. 3. In order of operation, the following grammar rules were applied for each manipulator:

A. Egg manipulator.
Palm: Rp1, Rp5, Rp4, Rp7, Rp5, Rp4, Rp7, Rp7.
For each finger: Rf3, Rf9, Rf10, Rf8, Rf12, Rf10, Rf8, Rf12, Rf11.
After developing all fingers: Rf2.

B. Wing screw manipulator.
Palm: Rp2.
Finger: Rf3, Rf5, Rf14, Rf14, Rf15, Rf2.

C. Water bottle manipulator.
Palm: Rp1, Rp6, Rp6, Rp5, Rp7, Rp4, Rp5, Rp5, Rp7, Rp7, Rp7.
Upper (abduction/adduction) finger: Rf3, Rf5, Rf14, Rf14, Rf14, Rf15, Rf2.
Lower (flexion/extension) fingers: Rf3, Rf7, Rf5, Rf14, Rf14, Rf14, Rf15, Rf2.
After developing all fingers: Rf2.

D. Scissor manipulator.
Palm: Rp1, Rp6, Rp5, Rp6, Rp7, Rp6, Rp7.
For each finger: Rf3, Rf7, Rf8, Rf12.
After developing all fingers: Rf2.

APPENDIX II
CAGE DEFORMATION: HIGH AND LOW RESOLUTION MESHES

Multiple high and low resolution meshes are involved in the design pipeline. Specifically, there are three meshes (shown in Fig. 3) associated with each grammar component:

- A cuboid, low-resolution “cage” mesh fully enclosing each grammar component. The vertices of this cage are used to define the basis of the deformation, and the user moves the cage vertices to deform the robotic manipulator components. These points are the only points the user has control over when altering the geometry of the manipulator: all other meshes deform according to the user-specified cage mesh deformations.

- A high-resolution mesh of each grammar component. Once combined to form a manipulator, these meshes will be 3D printed. This mesh is affected by deformations the user applies to the cage: when the user widens one end of the cage, the corresponding end of the high-resolution grammar component mesh will also widen.

- A coarse, low-resolution mesh used to generate knitting patterns. This is a highly simplified version of the grammar component and is sized so that the knitted pattern generated from this mesh fits snugly over the corresponding 3D printed part. These meshes are also subjected to the deformations the user applies to the deformation cage.

Special considerations beyond those listed above had to be taken in determining the dimensions of the low resolution cages for each component. Let the face of a grammar component that connects to another grammar component be called the mating face. Then, for every two grammar components that share a mating face, the vertices of the mating faces corresponding to those components must match for their two cage deformation meshes. This concept is illustrated in Fig. 10. Because the cage meshes must fully enclose all components and because cuboid meshes must always have aligning vertices, the height of the mesh must correspond to the largest component in the system. Similarly, the vertices of the mating faces of the tactile meshes of two components must also match. However, these tactile meshes are not required to fully enclose the high resolution mesh, nor are they required to be perfectly cuboid.

Why must this vertex-matching constraint exist? For the cage deformation mesh, the vertices of two mating faces must match because, upon mating, the overlapped vertices merge into one point that controls the two components on either side of the point. This ensures that the interface between the two components and any feature that spans those pieces remains intact. This principle guarantees both a watertight mesh and manufacturability after deformations. For the knitting mesh, the vertices of the two mating faces must align to form one continuous knitting surface. If the two faces did not match perfectly, there would be a sort of “step” between the smaller
and bigger faces, causing the knitting program to generate a cover with a step and corners that does not reflect any feature of the high-resolution mesh. To maintain smooth continuity, the two mating faces must perfectly align.

**APPENDIX III**

**CONTROL: FINITE STATE MACHINE**

Finite state machine controls based on sensor inputs were used to perform the manipulator tasks. These controls are not novel and were only intended to demonstrate that the manipulators are easily controllable; therefore, the positions of objects in the tasks were hard coded rather than determined intelligently via, for example, computer vision. In each of the state machine diagrams (Figs. 11-14), $p_{max}$ refers to the maximum reading in a sensor patch on a manipulator finger.

![Fig. 11. Finite state machine diagram for egg picker control sequence.](image)

**APPENDIX IV**

**MANIPULATOR PERFORMANCE**

This section contains image sequences with sensor data for each manipulator and task (Figs. 15-18). For each task, the threshold required to perform the task was experimentally determined. For instance, for grasping tasks, the threshold was chosen to correspond to the strength of grasp such that the object does not slip out of the hand. Please note that the sensor data plots are a depiction of a single run and are intended as only an approximate illustration of sensor responses while executing tasks.

![Fig. 15. A typical egg picking action sequence with the maximum readings from the tactile sensors on each finger. The grasp registers as successful if three out of the four finger readings exceed our set threshold.](image)

![Fig. 16. A typical wing screw tightening action sequence of the manipulator with the maximum readings from the tactile sensors on the finger. The wing screw is registered as tightened if it exceeds our set threshold.](image)
Fig. 17. A typical bottle task action sequence of the manipulator with the maximum readings from the tactile sensors on each finger. The water bottle registers as full if the pressure on either lower finger exceeds the threshold.

Fig. 18. A typical cutting action sequence of the manipulator with the maximum readings from the tactile sensors on the manipulator's thumb. The cut registers as “completed” if the pressure exceeds the threshold even if the scissors are not closed; this protects the motors from overload.