Sawyer Mobile Device Interaction

Aharon Sebton
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York, USA
ams9265@rit.edu

***Abstract*—Making collaborative robots touch device accessible is a relatively new and unexplored field. This paper proposes a new end effector and scripts to allow the Rethink Robotics cobot Sawyer to perform tapping, swiping, and pinching gestures on a stationary touch-enabled device. The successful efforts presented in this paper enable a future of cobots performing automated application/phone tests, or teleoperations using biological signals as inputs.**

***Index Terms*—cobots, touch screens, device testing, touch gestures, Sawyer**

I. Problem Statement

Collaborative robots, also known as cobots, are meant to safely perform given tasks while in the vicinity of humans working around, or with, them. The tasks performed by cobots also tend to be either tasks that humans are unable to perform safely, or tasks that are repetitive and capable of being automated. The effort displayed in this paper is an attempt to enable the Rethink Robotics cobot Sawyer to safely perform single and multi touch gestures on a touch-enabled device such as a smartphone or tablet. The eventual goal of this effort is twofold. One goal is to allow for the automation of application testing and device testing through the use of collaborative robots, rather than their more expensive industrial counterparts. The other goal is to allow for the teleoperation of collaborative robots using the classification of biological human signals, so that people with disabilities such as arthritis or muscular dystrophy can control their smart devices in a natural manner.

Enabling Sawyer to safely perform touch gestures on touch-enabled devices calls for multiple problems to be solved, since the robot was not designed with mobile device interaction in mind. The first problem is designing and manufacturing a new end effector for Sawyer. Because almost all modern day smartphones and tablets use capacitive touch interfaces, the part of the end effector that makes contact with the mobile device must be made of a soft, conductive, and grounded material. The surface area of the end effector that makes contact with the mobile device should also approximate that of a human finger. The end effector design should also be robust enough to withstand forces exceeding what is necessary to have a touch-enabled device recognize that a gesture is being performed. Finally, the end effector design should allow for both single and multi touch gestures, which means that one "finger" of the design should be able to make contact with the touch-enabled device while the other "finger" does not.

The second problem is controlling the amount of force that Sawyer applies to the touch-enabled device. The force applied must be enough for the device to recognize that it is being interacted with. The force applied must also be limited so that Sawyer does not damage or break the screen of the device it is trying to interact with.

The third problem is fixing the touch-enabled device in a stationary, upright position so that Sawyer can easily interact with the device and repeatable courses of action can be tested. To prevent the mobile device from sliding during testing, the lateral forces applied to the device by Sawyer during a "swipe" action should be counteracted by either a blockage or the frictional force between the device and the surface it rests on.

II. Literature Survey

A. Historical Solutions

Initial research was conducted to determine the most suitable applications for Sawyer to interact with modern mobile devices. The most notable applications involved robot teleoperation and smartphone testing.

Robot Teleoperation — Robot teleoperation (also referred to as telemanipulation) is a real-time method of a human controlling a robot from a safe distance, and it is often used "in scenarios where automation is impractical, where human judgement is essential, or where having the user engaged in the task is desirable", as explained by Rakita, Mutlu, and Gleicher [1]. Many novel ways of achieving teleoperation of arm robots have been developed in the past decade. Rakita, Mutlu, and Gleicher [1] developed a method of retargeting the motions of the Universal Robotics UR5 arm robot. Real world limitations during teleoperation include workspace size, motion smoothness, positional accuracy, joint constraints, and singularity avoidance during trajectory planning. Scaling factors are used when the size of the human user's workspace differs from the robot's workspace size. Hand position and orientation are used in conjunction with robot architectural constraints to perform inverse kinematics (IK) calculations, which find the set of robot joint angles necessary to achieve a given end effector position and orientation. Smoothness and accuracy terms are also used to determine the best robot trajectory possible when moving from its initial state to its desired state. Yang et al. [2] achieved the teleoperation of the Rethink Robotics cobot Baxter through IK, workspace mapping, and the acquisition and processing of human electromyography

(EMG) signals. This effort held promise for the ability to achieve stable teleoperation of Sawyer, since both cobots are manufactured by the same company, and their user interfaces are very similar. Ju et al. [3] began the teleoperation testing of Baxter using *forward kinematics* (FK) and a haptic feedback stylus. Forward kinematics is used for a robot to determine what set of translational and rotational motions will allow it to move the end effector from one "frame" (position and orientation) to another frame. Because the process of using FK to achieve human mimicry is iterative, the force generated by the haptic feedback stylus was made proportional to the amount of error between the human's new frame and the robot's current frame. In 2019, Yaovaja and Klunngien [4] managed to teleoperate an industrial robot over a large distance (over 200 km) using a high-speed 5G network. This could theoretically be accomplished with Sawyer, which would allow other research schools or companies to conduct their own testing with RIT's robot in the future. While the previously mentioned papers based their algorithms on the use of a stylus on a regular pad or surface, Li et al. [5] used a capacitive touch screen to introduce depth into the equation with their calligraphy robot. For devices with screens that are capable of measuring the amount of force applied to them (such as Apple devices with 3D Touch technology), the level of pressure applied to the screen and small fluctuation in end effector position affect the surface area of a "brush stroke" when drawing on the screen. 3D Touch and similar algorithms are also used in modern smartphones and tablets to introduce additional functionality.

Smartphone Testing — Smartphone automatic test systems are relatively new, and the field is currently dominated by expensive, custom-made industrial robots paired with some sort of computer vision subsystem. Juang, Tsai and Fan [6] applied Optical Character Recognition (OCR) techniques to overhead images of their test device, and used the results to detect the button or "object" the robot needed to reach to accomplish a task. They also used fuzzy logic to control the exact coordinates given to the robot that would ensure the button or object was pressed. Hsu, Lu and Juang [7] created a back propagation (BP) neural network (NN) that was able to generalize the coordinate conversion problem between camera and robot, and recognize patterns for more accurate object/button detection. Juang and Cheng [8] also use OCR techniques, in addition to pattern matching and edge detection, to assist in determining desired robot frames. The servo motors for each robot joint were controlled using proportional-integral-derivative (PID) controllers. Hsu, Lee and Shieh [9] introduced a novel adaptive GUI testing (AGT) method that allows for automatic testing of mobile applications, even if the application appears different across different device platforms and after applications are updated. This method also appeared to be the most capable of exhaustive application testing without the user hard-coding every possible sequence of actions within the app.

Other Applications — Ju et al. [10] applied a hybrid position/force controller using fuzzy logic to a robotic system designed for the rehabilitation of patients with neuromuscular disorders. Although this application is not something that could be easily implemented with a patient coming into physical contact with Sawyer, the position/force controller with fuzzy logic displayed here could be used with progressive re-mapping of workspaces to allow for other forms of rehabilitation applications.

B. Borrowed Methods

Chou and Juang [11] use a Denavit-Hartenberg (D-H) model of their five-degree-of-freedom arm robot to translate desired coordinates to a set of desired joint angles. They then use fuzzy theory to obtain coordinate errors and make minor adjustments. While the use of fuzzy theory is out of the scope for this effort, my completed robotics coursework at RIT enables me to extract a D-H table/model from a given arm robot architecture; I can do this for Sawyer. Timpea, Cosma and Sosdean [12] created their own custom end effector with three "fingers" to enable a robot to perform all types of touch gestures or "working movements". The design was meant to be mountable to any robot arm. Besides borrowing structural ideas from this paper, I can use the capacitive touch technology explanation presented here to my advantage during my own end effector design process. Serwadda et al. [13] created a Lego Mindstorms robot with a custom-designed robotic finger meant for the testing of touch-based authentication systems that I can use as inspiration for my own finger design. They also showed how the spatial distribution of swiping activity can be modeled as a 2D Gaussian, and provide a robust swiping algorithm based on this premise that I can employ. Liu et al. [14] designed and tested an optimal mechanism that applies constant force independent of displacement. Constant applied force is the ideal situation for interaction with expensive touch-enabled devices, especially if I conduct testing on my personal smartphone. This design's prototype was 3D printed, and I expect my end effector design to be 3D printed as well. Pan et al. [15] focuses on its OCR algorithm for testing mobile applications, but the use of this algorithm is out of scope for this effort. Instead, the simple and elegant coordinate system conversion between overhead camera and robot may be used to choose where Sawyer should practice its gestures. Kanstrén et al. [16] use Markov models to describe user behavior and create user profiles, which could be used in conjunction with the Gaussian swipe distributions in [13] to create custom Sawyer testing procedures before a specific user tries to perform a teleoperation. Craciunescu et al. [17] designed and manufactured their own delta robot capable of performing smartphone testing, and their end effector includes three separate fingers, two of which are controlled by stepper micro motors. If I choose to build an end effector with three fingers, using a stepper micro motor may give me more a consistent force applied to the screen I am testing on. Finally, Mao, Harman and Jia [18] built a custom robotic arm that is controlled using a Python IK script. It is likely that such an IK Solver script exists for Sawyer, whether it was made and distributed by Rethink Robotics or another member of the

robotics community. This should help me test which method is best for Sawyer to perform touch gestures in terms of repeatability.

III. Method

A. Proposed Approach

In order to solve the problem of a new end effector for Sawyer, I propose two ”fingers” that can be designed in CAD software and 3D printed. These fingers should have the ability to slide onto and lock with the ends of Sawyer's default ”electric parallel gripper” sets. Since Sawyer comes with multiple electric parallel gripper sets of different spans, there should not be any problem achieving both single and multi touch gestures on the same device. One finger should be placed out of the way of the smartphone or tablet while the other interacts with the device. The ends of the fingers should also be designed to fit small stylus pen tips on them. Touch-enabled devices should respond better to the stylus pen tip's conductive, rubbery material than to the polylactic acid (PLA) plastic used by most 3D printers. PLA was proven to be strong enough to withstand the forces applied to a screen by [14]. Using the insight about capacitive touch interfaces provided in [12], I propose taking a more passive approach than the battery-powered finger presented in [13]. Running a wire from each stylus pen tip to an electrically charged surface, such as Sawyer's endpoint or the metal surface of Sawyer's workspace, should help the capacitive touch screen to recognize that gestures are occurring. No external power source will be necessary for this design to be successful, and a third finger is not necessary to achieve the goal of performing the basic, commonly used touch gestures: tapping, swiping, and pinching. Opening and closing the existing gripper of Sawyer should help with pinching in and out.

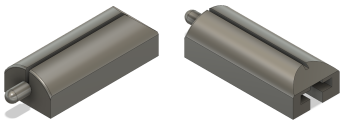
To solve the problem of force control, Sawyer already has force and torque-sensing capabilities built-in. The challenge here will be performing force analysis of the robot if the force applied by the end effector is unavailable to me directly. This should not be an issue as I have knowledge of, and experience performing, dynamic analysis of arm robots. If the necessary force data is already being sensed, I will learn how to read said data using a custom Python script and the robot operating system (ROS). Force data may also be available to read directly through Sawyer's Intera software. In order to control force applied to the screen, I will set a threshold to be reached as the finger(s) move toward the screen. Sawyer should stop moving its end effector towards the screen once this force limit has been reached.

Finally, to solve the problem of fixing a touch-enabled device is trivial for Sawyer's current workspace in RIT's Century Mold lab. A flat, raised metal surface is positioned directly in front of Sawyer, and the surface has holes for bolts to fit in them. A smartphone or tablet can be placed on this frictional surface and held in place by surrounding bolts.

B. Expected Results

The completion of semi-demos went smoothly. The first task completed was accessing Sawyer's force data. Sawyer was rebooted into Intera SDK mode, which allows it to be accessed by an in-lab computer over LAN connection and through ROS. Rethink Robotics has plenty of documentation on setting up a development environment and using the intera_interface Python library to access and change robot variables. The Limb class has a dictionary of forces and torques named joint_efforts, as well as a dictionary of forces and torques applied at the endpoint of the robot named endpoint_effort. The latter was used in a custom Python script to extract the force applied by the endpoint in the z direction, which is the direction the ”fingers” of the end effector will be pointing. I expected to be able to use this information to move forward with a script that would make Sawyer's end effector move toward the screen until a force threshold is reached (and therefore sufficient contact with the screen is made).

The second task completed was creating a CAD design of the finger that would be attached to the existing gripper. This was accomplished using the Autodesk Fusion 360 CAD software, as seen in Fig. 1. I based the bulk of the design off of a rubber part that came with Sawyer and was already designed to slide onto the electric parallel gripper. I chose a pack of stylus pen tips to order online (see Fig. 2), and added a peg to the end of my finger design that the stylus pen tip could fit over. Finally, I added grooves around the base of the peg and across the top of the finger to allow room for a grounding wire. I expected the 3D print of this part to go smoothly, and for the part to fit snugly with the existing gripper and stylus pen tip since I added small tolerances to the design.



(a) Front (b) Back

Fig. 1: End effector finger CAD model

Fig. 2: Stylus pen tip

The final task completed was integrating the touch interface with the table/Sawyer's workspace. This task was as simple as deciding where to screw bolts into the surface so that a permanent smartphone position and orientation was established. I expected my phone to remain stable and static throughout the testing process moving forward.

IV. Results

The fingers for the new end effector printed almost flawlessly. The part fit onto the electric parallel gripper snugly, and the stylus tips fit around the pegs snugly as can be seen in Fig. 3. The grooves I had added to hold the grounding wire were not wide or deep enough. Careful use of a band saw allowed for the adjustment of these grooves. The ends of wires were stripped and wrapped around the base of the peg of each finger before they were glued to each finger. The stylus pen tips were glued to the pegs of the fingers, guaranteeing contact with the exposed part of each wire as seen in Fig. 4. After sliding each finger assembly onto the electric parallel gripper of Sawyer, the final assembly step was electrical taping the other ends of each wire to a conductive section of Sawyer's endpoint. See Fig. 5 for an image of the final end effector assembly.

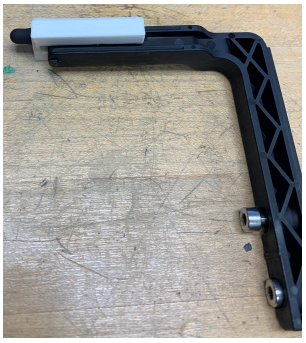


Fig. 3: Finger, tip, and gripper assembly

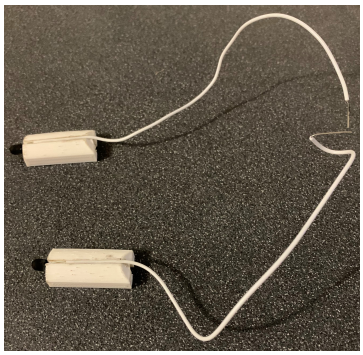


Fig. 4: Finger, tip and wire assembly

After an extensive exploration of possibilities using Sawyer's SDK and intera_interface Python library, I found that it was possible to record the robot's current configuration either by a set of joint angles or by the position and orientation of the endpoint. It was also relatively straightforward to command Sawyer to revisit these recorded configurations, without any D-H table, FK or IK analyses necessary. The Sawyer SDK even had an IK service client script made available if necessary. In order to create example scripts of taps, swipes and pinches, repeatable robot configurations were chosen by me using Sawyer's training cuff. A sequence

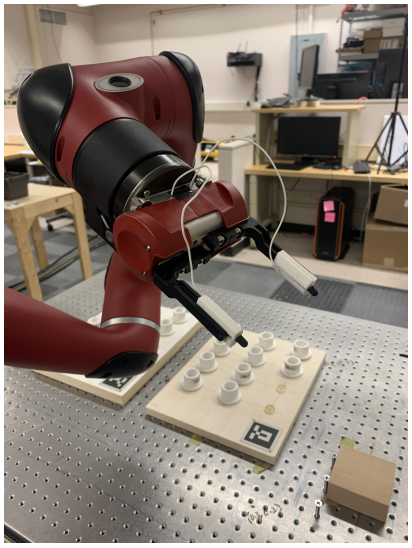


Fig. 5: Full end effector assembly

of frame transformations was chosen so that Sawyer's end effector hovers above the device under test (DUT), moves straight down to make contact with the device's screen, and moves back up after performing the task at hand. If a "tap" gesture was requested, the end effector moved back up almost immediately after making significant contact with the screen. If a "swipe" gesture was requested, the end effector moved in a chosen direction that was parallel to the plane of the screen before lifting back up. If a "pinch" gesture was requested, the gripper was opened or closed before the end effector lifted back up. Linear end effector motions were achieved using IK solution and joint trajectory solutions. An app designed specifically for device/application testing, Touchscreen Test by Vishal Singh, was purchased and used for testing these gestures.

One of the initial goals of this effort was to actively sense and check the force applied to the screen by the end effector, and to ensure that this force did not exceed a threshold to maintain the integrity of the DUT's screen. My literature review was unsuccessful in determining a safe force or pressure value to use as a threshold for my gesture algorithms. Therefore, I used Newton's third law of motion to my advantage. I brought Sawyer's arm to a "hover" configuration with the training cuff and brought my smartphone up to the stylus pen tip of one of the end effector's fingers. I pushed the smartphone up against the pen tip with more force until the phone became responsive to the contact force, then recorded the endpoint force value sensed by Sawyer at that time to determine a safe threshold. I was also unsuccessful in finding a way to implement an interrupt service routine (ISR) in my Python scripts. Therefore, I was only able to check the force sensed by Sawyer after completing the motion of the end effector down to make contact with the DUT's screen. If the force in the z direction exceeds the threshold defined, Sawyer's limbs are disabled and the motion routine is exited.

V. Conclusion and Future Work

A. Conclusion

A simple and robust end effector for mobile device interaction was developed and integrated with the Sawyer cobot. Single and multi touch gesture routines were implemented in Python using inverse kinematics and joint trajectory planning solutions provided by Sawyer's SDK. Force data was accessed at the endpoint of the end effector, but it could not be checked continuously through the Sawyer SDK system. All gestures were still performed by Sawyer safely and successfully, and the visual residues from the smartphone testing application suggest that the gesture scripts created had a high level of repeatability, but this has not been tested in a quantifiable manner.

B. Future Work

It was later discovered that force data can be read and compared to a given threshold using Sawyer's default Intera software. The first and most important future effort will be to implement more active versions of my gesture routines, so that Sawyer truly stops moving down towards the screen once the set force threshold is met. Following this, I would like to run repeatability tests and compare the results using Sawyer to similar test results with industrial robots interacting with mobile touch-enabled devices. I would also like to explore the generalization of each gesture so that device/application testing can further approach human-like gesture behavior; this could be achieved using the two-dimensional Gaussian distributions of swipe activity found in [13]. I would like to add computer vision capabilities to the system, either using Sawyer's built-in arm camera or an overhead Intel RealSense RGBD camera. My literature review proved text and button recognition to be a useful tool in the world of automated device and application testing. Finally, I would like to achieve the teleoperation of Sawyer through collection and processing of biological signals such as EMG, gaze directions, and gaze fixations.

References

[1] D. Rakita, B. Mutlu, and M. Gleicher, "A Motion Retargeting Method for Effective Mimicry-based Teleoperation of Robot Arms", 2017.

[2] C. Yang, S. Chang, P. Liang, Z. Li, and C.-Y. Su, "Teleoperated robot writing using EMG signals", 2015.

[3] Z. Ju, C. Yang, Z. Li, L. Cheng, and H. Ma, "Teleoperation of humanoid baxter robot using haptic feedback", 2014.

[4] K. Yaovaja and J. Klunngien, "Teleoperation of an Industrial Robot using a Non-Standalone 5G Mobile Network", 2019.

[5] J. Li, W. Sun, M. Zhou, and X. Dai, "Teaching a calligraphy robot via a touch screen", 2014.

[6] J.-G. Juang, Y.-J. Tsai, and Y.-W. Fan, "Visual Recognition and Its Application to Robot Arm Control", Applied Sciences, vol. 5, no. 4, pp. 851–880, 2015.

[7] W.-T. Hsu, C.-C. Lu, and J.-G. Juang, "Application of the LM-HLP Neural Network to Automatic Smartphone Test System", 2018, pp. 217–225.

[8] J. Juang and I. Cheng, "Application of character recognition to robot control on smartphone test system," Advances in Mechanical Engineering, vol. 9, (3), 2017.

[9] C.-W. Hsu, S.-H. Lee, and S. W. Shieh, "Adaptive Gestures for GUI Testing on Smartphones", IEEE Software, pp. 1–1, 2015.

[10] Ming-Shaung Ju, C.-C. K. Lin, Dong-Huang Lin, I.-S. Hwang, and Shu-Min Chen, "A rehabilitation robot with force-position hybrid fuzzy controller: hybrid fuzzy control of rehabilitation robot", IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 13, no. 3, pp. 349–358, 2005.

[11] Kuan-Yu Chou and Jih-Gau Juang, "Automatic Dialing Test System Based on Visual Servo Control", Sens. Mater., Vol. 33, No. 8, 2021, p. 2773-2788.

[12] S. Timpea, C. Cosma, and D. Sosdean, "Touch Screen Tester Device End-Effector", Materiale Plastice, vol. 56, no. 2, pp. 445-448, 2019.

[13] A. Serwadda, V. Phona, Z. Wang, R. Kumar, and D. Shukla, "Toward Robotic Robbery on the Touch Screen", ACM Trans. Inf. Syst. Secur. 18, vol. 4, no. 14, 2016.

[14] C.-H. Liu, M.-C. Hsu, T.-L. Chen, and Y. Chen, "Optimal Design of a Compliant Constant-Force Mechanism to Deliver a Nearly Constant Output Force Over a Range of Input Displacements", Soft Robotics, vol. 7, no. 6, 2020.

[15] Z. Pan, J. Chen, L. Yao, and Z. Chen, "Research on Functional Test of Mobile APP Based on Robot", 2020.

[16] T. Kanstren, P. Aho, A. Lamsa, H. Martin, J. Liikka, and M. Seppanen, "Robot-assisted smartphone performance testing", 2015.

[17] M. Craciunescu, S. Mocanu, C. Dobre, and R. Dobrescu, "Robot Based Automated Testing Procedure Dedicated to Mobile Devices", 2018.

[18] K. Mao, M. Harman, and Y. Jia, "Robotic Testing of Mobile Apps for Truly Black-Box Automation", IEEE Software, vol. 34, no. 2, pp. 11–16, 2017.