23137833

Spring-Based Haptic and Texture Feedback Control System

Wisnu Prasetya Mulya

Abstract—**The use of robotic systems in surgeries is becoming more prevalent. Currently, surgeons rely on the robot's visual feedback of deformed tissues in contact with the instrument to infer how much force is applied. Haptic and texture feedback for medical robotic systems is a developing field that can improve surgeons' performance and patient outcomes. This paper aims to develop algorithms or control systems that can give the robot operator haptic and texture feedback based on the interaction with virtual environments. The first step in establishing the system was modelling the force control by a spring-based dynamic system, followed by haptics feedback on a virtual bounding box, and finally, the development of texture feedback on a virtual wall. By the end, the project successfully established a spring-based force control system by simulating a spring-like movement and the haptic and texture systems by simulating sensations to the operators working in a virtual environment.**

***Index Terms*—control, dynamics, force, haptics, kinematics, robotics, texture.**

I. INTRODUCTION

THE field of medical robotics has seen significant growth in recent years. The growth in the number of medical and engineering publications in medical robotics, starting from six publications in 1990, has increased to more than 3500 publications by 2020 [1]. Of these publications, 60% to 70% of them are dominated by medical papers on laparoscopic robots, which is influenced by the success of the Intuitive Surgical's da Vinci Surgical System. Considering the size and growth of the surgical robotics field, any development or research focusing on solving real-world problems in the field could potentially gather the attention of fellow researchers and engineers to work on solutions in a collaborative and effective manner.

One of the problems in this area is the meagre number of robotic systems that can relay haptic or tactile feedback [2]. With such feedback, surgeons can tell how much force they apply to the organs they are operating. Further, haptic or tactile feedback can help clinicians diagnose malignant tissues from normal ones by the difference in softness [3].

The current solution uses visual feedback to substitute surgeons' sense of touch [2]. Surgeons can tell how much force they apply to tissues with the robotic instrument by looking at the deformity of tissues in contact with the robot.

This paper aims to address the issue of the lack of touch sensation by developing algorithms or control systems to simulate haptic and texture feedback for a robotic system operator. Building a simple and reusable haptic and texture feedback algorithm that can be implemented in typical robotic systems would enable the development of the same algorithm or system, specifically in medical robots.

Virtual environments were used in this paper to simulate interactions of an operator using the robotic system with virtual objects. The main idea was to have the operator perceive a believable touch sensation from the robotic system when interacting with objects in the virtual environment. This is similar to when surgeons operate with surgical robotic systems, which only have visual feedback from the robots interacting with the patients.

The development of the algorithm or system was divided into three main milestones. The first was creating a force control algorithm to simulate a spring-like movement. This force control would be the backbone of further force control to simulate haptic and texture.

The second milestone was building an algorithm to simulate a haptic sensation from the operator's interaction with a bounding box object in the virtual environment. The algorithm built in the first milestone was used and extended to develop the touch sensation so that the robot operator can feel the sensation of touching an object and tell the shape of it.

The third milestone was to use and extend the previous algorithm further by adding texture sensations. A virtual object was also created for the operator to be able to interact with and feel the texture of it. The operator should be able to tell whether the object is smooth or rough.

All these milestones culminated in an objective to build several algorithms that are simple and easy to replicate for the implementation of a haptic and texture feedback feature in robotic systems. The outputs of this paper could potentially be further developed and implemented to fit any specific robotic system and task, such as medical robotic ones.

II. METHODOLOGY

A. Experimental Design

The first experiment was the force calibration of each of the motors' current readings. This calibration determined the relationship between the motors' currents and the torque they apply at the robot arms' ends. This step was necessary since the output of the control algorithm needed to be in terms of motors' currents. Another reason for obtaining this relationship was to measure the torque applied at the robot's tip.

Further, three milestones were developed incrementally to achieve a haptic and texture feedback algorithm. Each one of these milestones had a different experimental design:

1. Spring-based force control
2. Haptic feedback on virtual bounding-box
3. Texture feedback on virtual wall

For the first milestone, the paper focused on building a spring-based force control in a robotic system. This was done by having the robot's tip in a fixed home position and only moving it back to it when the tip was displaced. The movement of the robot trying to get back to its home-position was implemented by a force control system that simulated a spring-like movement.

Validating the first milestone involved in evaluating whether the movement of the robot's tip when displaced from its home-position emulates the movement of a spring when displaced from its equilibrium state. Some fine-tuning was done towards the constants of the formula for the force control to emulate the spring-like movement.

The second milestone was achieved by extending the algorithm created in the first milestone to give back haptic sensation in a robotic system. A two-dimensional virtual environment was set up to give visual feedback to the robot's operator on the location of the bounding box and the robot's tip. No force is applied to the operator whenever the robot's tip is within the box. However, when the tip is on the boundary of the box or beyond, there is a force acting on the robot to give a haptic sensation to the operator as if the robot is touching an object in the real world. Some adjustment was also applied to the force control formula to gradually increase the reaction force as the tip entered the box's wall.

There were two steps in validating the second milestone. The first was to compare the target torques with the measured output torque. Having this comparison would tell whether the algorithm can successfully give torque to the robot arms with the value that the simulation targets to have. Secondly, the paper conducted a trial with human operators, whose task was to tell the shape of the object they interacted with without seeing the virtual environment displayed. The success of this trial was determined by whether most of the human operators could tell the shape of the virtual object.

The third milestone extended the algorithm to give back texture sensation in a robotic system. The experiment of this milestone was conducted by first setting up a virtual wall object with texture from which the operator can interact by touching the wall with the robot's tip. Some adjustment to the force control system was done to simulate the sensation of a rough-textured wall.

The validation of this milestone also took two steps. They are similar to the validation from the second milestone, of which the first one was to compare the target torques and the measured output torques, while the second one was by a trial with human operators to evaluate whether they can differentiate a soft-textured wall with a rough-textured one.

B. Equipment and Setup

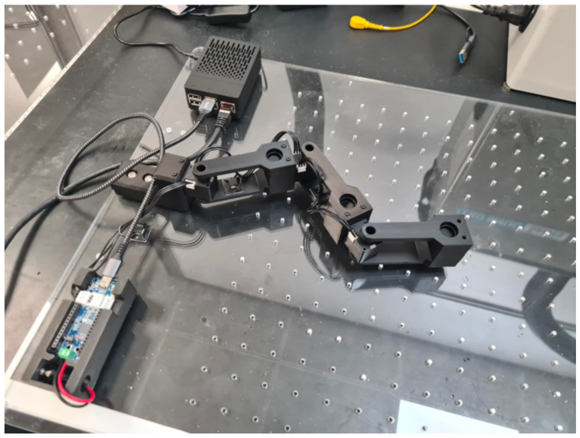


Fig. 1. The Robotic System Setup

The equipment and setup in this paper included hardware used for the robot and its calibration and software used for building the algorithm and visualising the virtual environment.

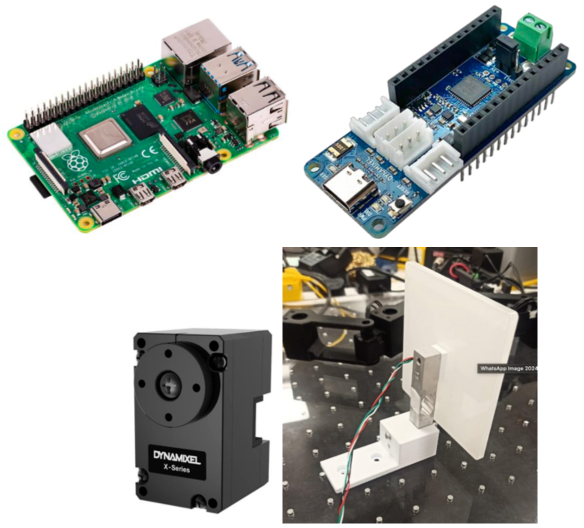


Fig. 2. The Hardware: (a) the top-left is a Raspberry Pi 4, which runs on a Linux operating system; (b) the top-right is an OpenRB-150; (c) the bottom-left is a Dynamixel XC330-M288-T, which is used for the robotic arms' motors; (d) the bottom-right is a force load cell which is used for the force calibration.

The robotic setup, as seen in Fig. 1, contained hardware equipment of Raspberry Pi and OpenRB-150, which are shown in Fig. 2(a) and Fig. 2(b), and three connected 3D-printed robotic arms, which are actuated by three Dynamixel motors shown in Fig. 2(c).

Each robotic arm has a length of 10 cm measured from the length of two motors' centres, while the last arm has a length of 6.75 cm measured from the motor centre to the tip's centre. When the arms are straightened, they have a length of 26.75 cm.

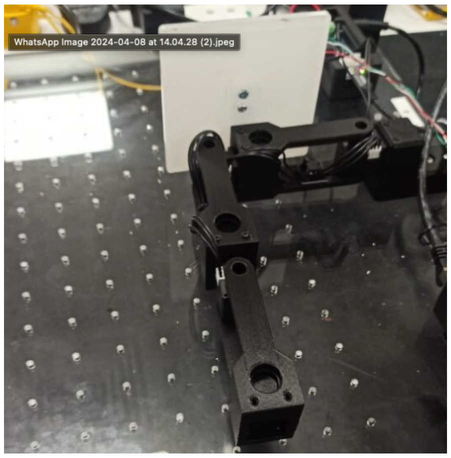


Fig. 3. The Force Calibration Setup

In addition, there was also a hardware setup regarding the force calibration procedure, as shown in Fig. 3. The force calibration used a force load cell, which is connected to a 3D-printed contraption that connects it to the robotic setup's acrylic base and an acrylic square plate from which a force is applied.

Further, the software used to develop the algorithms and as the base of the robotic operating system are ROS 2, Python 3, PyQt 5 Python library, and Microsoft Remote Desktop [4], [5], [6], [7].

The ROS 2 is installed in the Raspberry Pi as the operating software for the robotic system. The algorithms developed were written using Python 3, while the visualisation was rendered using PyQt 5 and displayed on the operator's computer system connected to the Raspberry Pi using Microsoft Remote Desktop.

C. Variables and Measures

For the force calibration step, each of the motors was calibrated with the relationship of the following variables:

$$F = k \times i \tag{1}$$

Where F is the force in Newton, k is the relationship constant between the motor's current and the force it exerts, and i is the motor's current. The k constant is obtained by finding the relationship between the current working on the motors and the force exerted on the load cell plate.

For the validation of milestone one, to evaluate the movement of the robot's tip when displaced from its home position, the variable to measure was the displacement in meters. This variable was measured by the length of the robot's tip to the home position.

Regarding the validation of torque comparison for milestones two and three, there were two variables to measure: target torque and measured torque. Target torque was obtained from the output of the control algorithm in torque, while the measured torque was obtained by estimating the current sensed from each of the motors.

Further, regarding the human operator trial for the second milestone, the variable to measure was the success rate of participants in guessing the shape of the bounding object. This was measured by the correct answer from the operator interacting with the bounding-box without using the visualisation of the virtual environment.

Lastly, regarding the human operator trial for the third milestone, the variable to measure was the success rate of participants in guessing the roughness of two virtual wall objects. The participants were given two virtual walls, one soft and the other rough, to interact with. A success in guessing the roughness is when the participant can tell which one is which.

D. Procedure

D.1. Force Calibration

In the force calibration process, the load cell itself needs to be calibrated first since the output of its reading is in a unitless raw value. The following equation is considered for the load cell calibration:

$$F = (R\text{-}SS) \times k \tag{2}$$

Where F is the force in Newton, R is the raw value read from the load cell, SS is the steady-state constant of when the load cell plate is in the vertical position, and k is the relationship constant of the raw value to force.

SS in equation (2) was measured by taking the median value of several readings from the load cell in a vertical position. After that, the k in equation (2) was obtained by comparing the reading from two weights of 50 g and 100 g. The difference between the two readings was then compared to the theoretical difference of weight, which is 0.49035 N.

Following the load cell calibration, each motor's current relationship to its force readings was calibrated. This process was conducted with the setup portrayed in Fig. 3. Currents from zero to 500 mA were sent incrementally to each motor and pushed against the load cell's plate. The readings from that process were then used to obtain the k constant in equation (1) for each of the motors by linear regression method.

D.2. Spring-Based Force Control

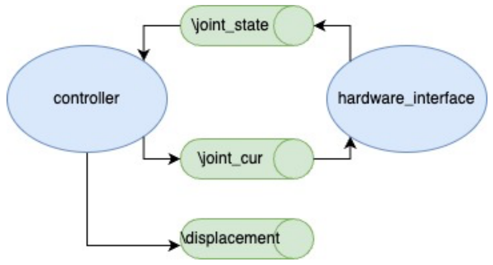


Fig. 4. ROS Architecture of the Spring-Based Force Control

Two primary ROS nodes were used to create the spring-based force control system as portrayed in Fig. 4. The first one was called *\hardware_interface*, which is used to read the joint positions and currents of the motors, which are sent to the *\joint_state* topic. The second is the controller node, where the spring-based force control algorithm runs. The output of this control system was the currents sent for each of the motors, published in the *\joint_cur* topic, and the displacement value of the robot's tip from its home position for the validation step, published in the *\displacement* topic.

Algorithm 1 Spring-Based Force Control

```
1. Get joint positions
2. Get the cartesian position of the robot's tip with
   forward kinematics
3. Calculate the robot's tip distance from its home
   position
4. If the distance is greater than tolerance, send current
   to motors calculated from force to get to the home
   position
5. Else, start from the beginning
```

The algorithm implemented in the controller node was generally portrayed in Algorithm 1. The calculated force was using the following equation:

$$F = k.dx - d.v + Tf + M(q, dq).ddq + c(q, dq) \quad (3)$$

Where F is the force in Newton, k is the spring constant, dx is the distance between the robot's tip and its home position, v is the robot's tip velocity, q is the joint positions, dq is the joint velocities, ddq is the joint acceleration, Tf is the friction in Newton, $M(q, dq)$ is the matrix inertia given joint positions and velocity, and $c(q, dq)$ is the centrifugal and coriolis value given the joint positions and velocity.

The terms Tf, $M(q, dq)$, and $c(q, dq)$ in equation (3) represent the dynamics of the robotic system. The friction relationship with the motor's velocity was obtained by finding the representative equation with a logistic regression method and relating it to the force afterwards. The following equation relates dq (joint velocity) with I (motor's current), and the function is portrayed in Fig. 5:

$$I = \frac{40}{1 - \exp(dq)} - 20 \quad (4)$$

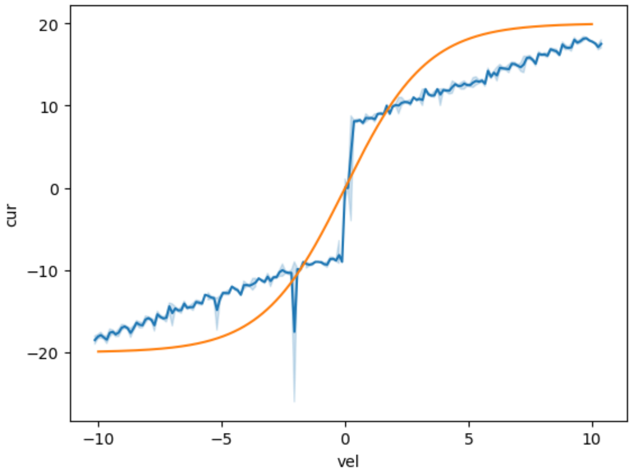


Fig. 5. The Plot of Joint Velocity to Current and Its Logistic Regression

The other dynamics terms from equation (3) were obtained by the Lagrangian Formulation approach [3].

Aside from the dynamics, the spring term k and the dampening term d were fine-tuned with multiple values of 15 and 30 for the k and 0.5 and 1 for the d.

D.3. Virtual Environment Visualisation

The virtual environment was visualised using a Python library, PyQt5, and displayed on the operator's computer using the Microsoft Remote Desktop application. The visualisation portrayed is that each of the robotic arms is centred at (0, 0) coordinates, a square box with the size of 15 by 15 centred at (0, 13.5) coordinates, and a vertical line intercepting the horizontal line at (0, 21) coordinates. The display was two-dimensional, with the vertical line going positive upward and the horizontal line going positive leftward.

The square box represents the bounding box where the robot's tip is confined, and the vertical line represents the wall used for the texture rendering.

D.4. Haptic Feedback on Virtual Bounding-Box

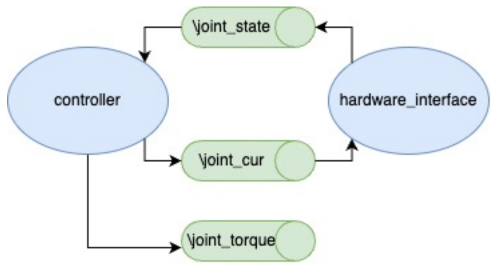


Fig. 6. ROS Architecture of the Haptic and Texture Feedback Systems

For the haptic feedback system, there were also two primary ROS nodes, just like with the spring-based force control. The main difference here is that the controller node published its output of target joint torques and calculated joint torques to a topic named *\joint_torque* for further analysis of the validation step.

Algorithm 2 Haptic Feedback on Virtual Bounding-Box

```
1. Get joint positions
2. Get the cartesian position of the robot's tip with
   forward kinematics
3. If the robot's tip is outside the bounding box, send
   current to motors calculated from force
   perpendicular to the box's wall
4. Else, start from the beginning
```

The algorithm implemented to the controller node in this milestone was generally portrayed in Algorithm 2. It is very similar to the Algorithm 1. There were two differences between them. The first one is the direction of the reaction forces sent to the robot's tip, which is to the home position for Algorithm 1 and perpendicular to the bounding box's wall for Algorithm 2. Also, the reaction forces at the box's corners are diagonally inwards, as portrayed in Fig. 7.

The second difference was with the calculation of the magnitude of the reaction force. Applying a reaction force with a magnitude from equation (3) resulted in highly unstable haptic feedback, such as a hard shakiness of the robot's tip at the box's wall and a very bouncy reaction when the tip enters the wall. A different strategy was implemented instead. A gradual increase of the *k* constant in equation (3) from 0 in the value set was implemented so that the hardness of the wall increases the further the distance of the robot's tip from the wall. This gradual increase was applied using the following equation (5) formula.

$$k_g = k.exp(40.d) - 1 \tag{5}$$

Where k_g is the new gradual *k* spring constant, *k* is the spring constant from equation (3), and d is the displacement length of the robot's tip from the box's wall.

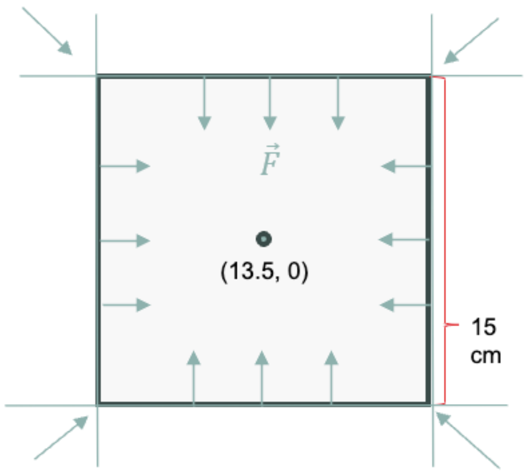


Fig. 7. The Bounding Box with the Direction of the Reaction Forces

D.5. Texture Feedback on Virtual Wall

The ROS architecture for the third milestone is the same as for milestone two, as portrayed in Fig. 6.

Algorithm 3 Texture Feedback on Virtual Wall

```
1. Get joint positions
2. Get the cartesian position of the robot's tip with
   forward kinematics
3. If the robot's tip is to the left of the wall, send
   current to motors calculated from force in the
   direction to the right of the wall, with different k and
   d constants from equation (3) based on the area in
   the wall.
4. Else, start from the beginning
```

As for the algorithm implemented in this milestone, portrayed in Algorithm 3, it is mostly similar to Algorithm 2, except with the direction of the reaction force direction, which is to the right of the wall. Another difference was also in the different coefficients for the reaction force, which was intended to simulate the sensation of the texture of the wall.

The difference in the texture was simulated by having a changing *k* constant from equation (3) at intervals for the reaction force, depending on the area of the wall the operator interacts with. A visualisation of this algorithm is displayed in Fig. 8. The wall was divided into intervals of width 0.5 cm with two kinds of constant: high and low. One interval has the *k* of 70 and *d* of 2, while the other has *k* of 10 and *d* of 2.

The interval width of 0.5 cm was fine-tuned by testing different widths from 0.25 cm to 5 cm. The width of less than 0.5 cm felt approximately less textured due to roughness that was too fine, while the width of greater than 0.5 cm felt not too different from the reaction force instability with the smooth wall.

For fine-tuning the *k*, it was tested from 10 to 100 without changing the damper constant *d*. Having *k* greater than 70 resulted in a too-strong reaction force, and the difference of 60 in value with the lower constant one felt the best for simulating a rough texture.

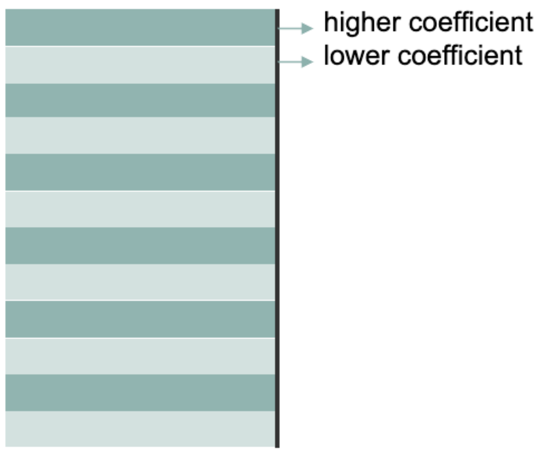


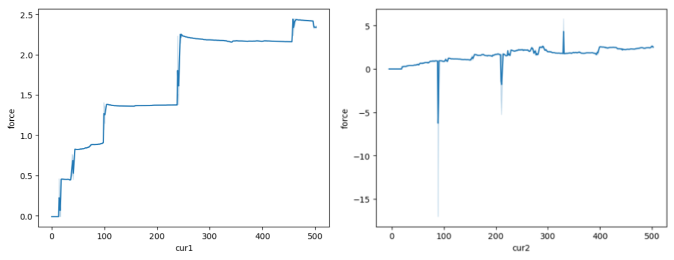
Fig. 8. The Diagram of the Virtual Wall Design with Interchanging Reaction Force Coefficients to Simulate Texture

E. Means of Result Collection

The variables to measure, including the force reading, the robot's tip distance, the target torques, and the measured torques, were collected from the published values in the corresponding topics using ROS 2's command of rosbag [8]. The raw rosbag files were then downloaded from Raspberry Pi and analysed in plots with Python using a package named rosbags [9].

III. Results

A. Force Calibration



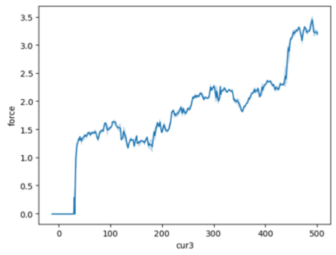


Fig. 9. The plots of motors' current to force: the top-left is for motor 1 (a), the top-right is for motor 2 (b), and the bottom is for motor 3 (c). The horizontal axis represents the current in mA, and the vertical axis represents the force in N.

To get the constant k from equation (1) for each motor using linear regression, we sampled only in the range of 100 mA to 200 mA for motor 2, 180 mA to 300 mA for motor 3, and all of the data for motor 1. This sampling was due to the volatile results, and the decided ranges were based on the area with a roughly positive linear trend, except for motor 1, where it seemed to increase in steps, so the whole data were considered.

Obtaining the gradient constants of linear regressions to each plot, the k for motor 1 was 0.0035 N/mA, the k for motor 2 was 0.1130 N/mA, and the k for motor 3 was 0.0066 N/mA.

B. Spring-Based Force Control

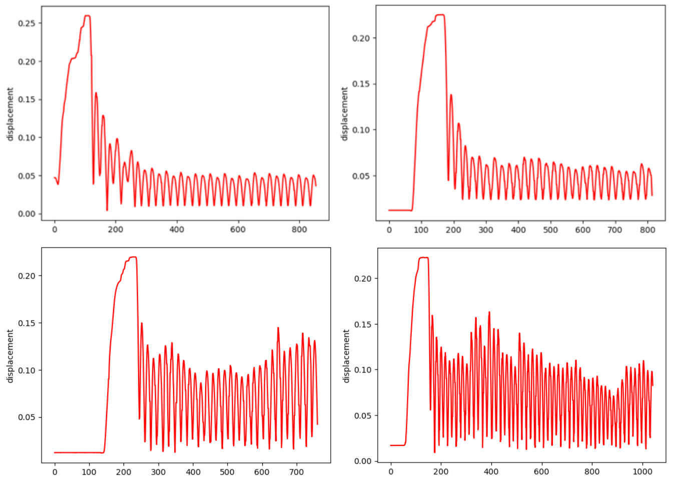


Fig. 10. The plots of the robot's tip distance to home position: the top-left is for k equal to 15 and d equal to 0.5 (a), the top-right is for k equal to 15 and d equal to 1 (b), the bottom-left is for k equal to 30 and d equal to 0.5 (c), and the bottom-right is for k equal to 15 and d equal to 0.5 (a). The horizontal axis represents the time, and the vertical axis represents the distance.

From a rough visual observation of all plots from Fig. 10, the oscillation of the robot's tip did follow the movement of a spring-damper in a one-dimensional axis, of which there is a large oscillation at the beginning and then a decrease in size.

Comparing the movement from the fine-tuning of the variables in equation (3), k was picked to be 15 for the algorithm since it gives lower oscillations over time and, hence, the stability of the robot's movement, compared to the value of 30. Further, the d picked was the value of 0.5 since it appeared to have less volatility compared with the one with a 1.0 value.

C. Virtual Environment Visualisation

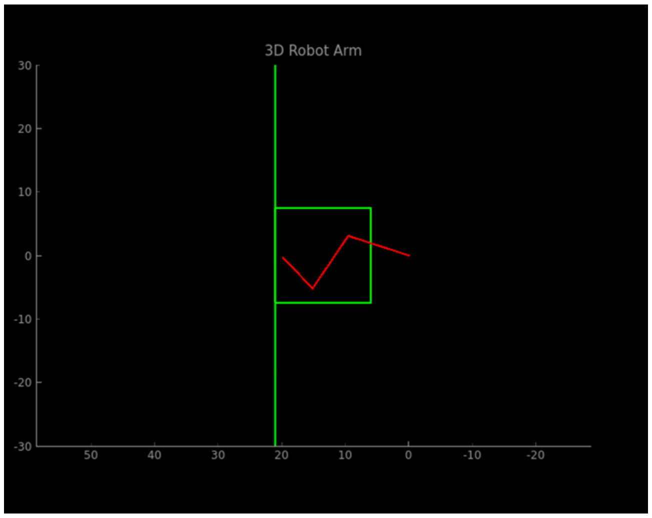


Fig. 11. The Visualisation of Robotic Arms, Bounding Box, and Textured Wall

All the visualisation for the robot's arms, the bounding box, and the textured wall were done simultaneously, as portrayed in Fig. 11. The vertical line representing the wall overlapped with the left side of the bounding box.

The rendering of the robot's arms was also very good in that there was perceptibly no lag between the physical movement of the arms in the real world and the visualised ones.

D. Haptic Feedback of Virtual Bounding-Box

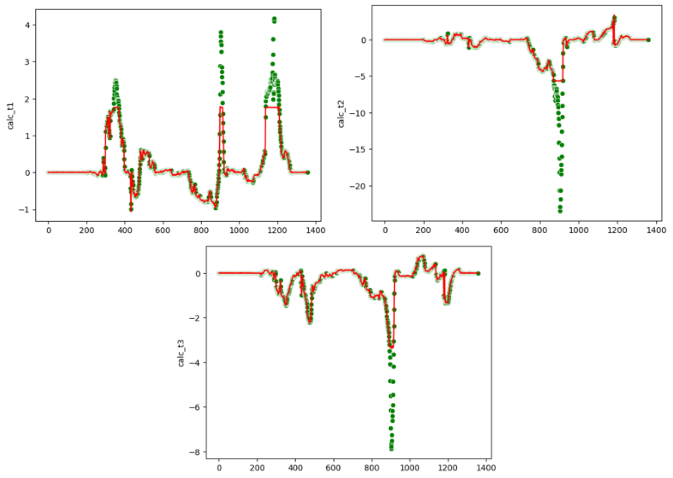


Fig. 12. The plots of motors' measured torques (in Red) to the target torques (in green) for bounding box evaluation: the top-left is for motor 1 (a), the top-right is for motor 2 (b), and the bottom is for motor 3 (c). The horizontal axis represents time, and the vertical axis represents torque.

With the result of the torques comparison portrayed in Fig. 12, the measured torques can be seen to follow the shape of the target torques. It suggests that the direction of the reaction force was simulated well. Further, the gap between the target and the measure torques was due to the maximum current imposed on the robotic system. This constraint made the robot unable to achieve the target torques.

Another evaluation of this milestone was with the human operator's trial. The result was an 80% success rate. Four of the five participants in the trial could correctly tell that the shape of the bounding object was a box or square.

E. Texture Feedback on Virtual Wall

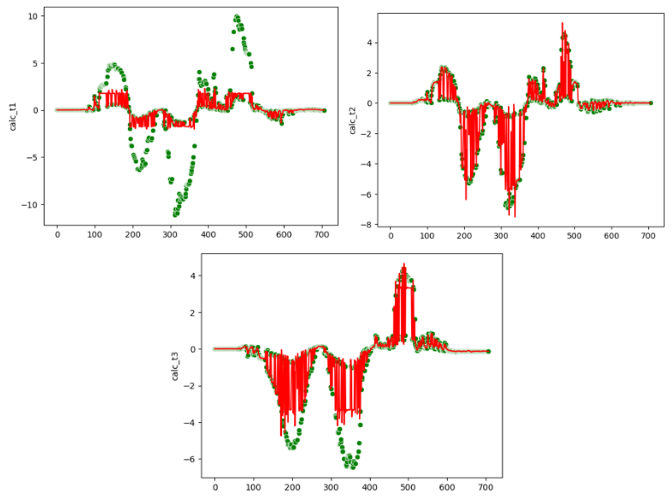


Fig. 13. The plots of motors' measured torques (in Red) to the target torques (in green) for textured wall evaluation: the top-left is for motor 1 (a), the top-right is for motor 2 (b), and the bottom is for motor 3 (c). The horizontal axis represents time, and the vertical axis represents torque.

From the Fig. 13, the overall shape of the measured torques followed the target ones well. It suggests that the direction of the reaction force was simulated well. The gap between the target and the measured torque was also due to the maximum current imposed on the robotic system, as with the haptic feedback on the bounding box. Also, the rapid oscillation shape of the measured force suggests that the sensation of texture was emulated well, too. This is because the texture is achieved by alternating the reaction force magnitude from low to high for every 0.5 cm distance of the wall.

Additionally, the result of the human operator's trial was a 100% success rate. Out of six participants, all could tell the difference between the rough and soft walls.

IV. Discussion

In summary, all the milestones in this project for developing haptic and texture control algorithms achieved their goals by simulating a spring-based movement, exerting torques that mimic the target ones, and giving haptic and texture sensations that are believable to human operators. These results were also obtained by using algorithms that are simple and easy to replicate in typical robotic systems.

Nevertheless, several limitations of this research need improvement. The first one is with the force calibration, of which the recorded data did not resemble a linear relationship with the motors' currents. This downfall limits the accuracy of calculating the currents when exerting a particular amount of force.

Further, there is a limitation with the validation of milestone 1's distance from home position over time, whose fine-tuning was done by a visual observation approximation. Especially for the case of picking the *d* constant from equation (3), the volatility in the oscillations between different values was too small to be sure which one was better.

Moreover, there are large observed gaps between the targeted and measured torques in the milestones 2 and 3 results. Even though the research successfully emulated the shape of the intended movement with torques, there were multiple instances where the target torques were not achieved.

Several things could be done to improve the limitations imposed on the research. With the issue of the calculated force accuracy, the robotic system would benefit from having dedicated force or torque sensors, which would be more accurate than the approximation with the motor's current.

Next, a statistical evaluation could address the issue of picking the best constants. With statistical evaluation, the difference between volatilities of the distance oscillation could be made sure by calculating whether the difference is significant.

With the gaps between the torques' limitations, future work could be done where the robotic system would have higher current constraints to achieve the target torques.

Finally, the outcome of this research gives multiple algorithms that can benefit from further developments. Fellow engineers and researchers could build packages or libraries in Python or C++ that would be suitable for typical robotic operating systems. Such development would prove helpful to people in the field who need simple and lightweight plugins for haptic and texture feedback in their robotic systems.

Acknowledgement

The author is deeply grateful to his research team's colleague, Estya Nadya Meitavany, for her resilience and intelligence in this project. The same thing is also owed to other colleagues from the same module, The Hung Dang, Farhana Moosa, and Jiaheng Wang.

Further, deep appreciation is also due to the lecturers and teaching assistants for the author's Advanced Robotics module at King's College London: Lukas Lindenroth, Alejandro Granados, Aya Zeidan, and Peiyuan Sun; with their intelligent and kind advice to the project.

References

[1] P. E. Dupont *et al.*, "A decade retrospective of medical robotics research from 2010 to 2020," *Sci Robot*, vol. 6, no. 60, p. 8017, Nov. 2021, doi: 10.1126/SCIROBOTICS.ABI8017/SUPPL_FILE/SCIROBOTICS.ABI8017_SM.PDF.

[2] N. Enayati, E. De Momi, and G. Ferrigno, "Haptics in robot-assisted surgery: Challenges and benefits," *IEEE Rev Biomed Eng*, vol. 9, pp. 49–65, 2016, doi: 10.1109/RBME.2016.2538080.

[3] B. Siciliano and O. Khatib, "Springer handbook of robotics," *Springer Handbook of Robotics*, pp. 1–2227, Jan. 2016, doi: 10.1007/978-3-319-32552-1/COVER.

[4] "Microsoft Remote Desktop - Microsoft Apps." Accessed: May 06, 2024. [Online]. Available:

https://apps.microsoft.com/detail/9wzdncrfj3ps?hl=en-us&gl=US

[5] "PyQt5 · PyPI." Accessed: May 06, 2024. [Online]. Available: https://pypi.org/project/PyQt5/

[6] "Download Python | Python.org." Accessed: May 06, 2024. [Online]. Available: https://www.python.org/downloads/

[7] "ROS 2 Documentation — ROS 2 Documentation: Rolling documentation." Accessed: May 06, 2024. [Online]. Available: https://docs.ros.org/en/rolling/index.html

[8] "ros2/rosbag2." Accessed: May 06, 2024. [Online]. Available: https://github.com/ros2/rosbag2

[9] "rosbags · PyPI." Accessed: May 06, 2024. [Online]. Available: https://pypi.org/project/rosbags/