Mohamad Sabagh, Muhammed Saad Qadri

ForceSense: Advancing Robotic Biopsies through Redundant Joint Feedback Integration

Project documentation

Further instructions

How to run our program is simple. Simply start the biopsy program by running ./biopsy.cpp 192.168.1.107
Refer to README.txt for more information about the registration process that needs to be done only once.

From here you will be greeted with a terminal prompt similar to PS2(options 1-3 are implementations from PS2). Options 4 and 5 are the ones of interest for our project. Option 5 will use a bitwise step implementation of our biopsy program that has a force feedback option on the input joint. Option 4 is then to run the impedance control implementation of our project which was the first version of the code.

On both options, you will be greeted with another prompt asking for the X, Y, and Z coordinates in voxel space. Once the correct values in accordance with where the tumor is in voxel space are inputted the robot should enter into a hovering state. After the hovering state, there will be a bunch of rows and columns of 0s which are reading threshold values set in setCollisionBehaviour and will print a 1 if contact or collision is detected. Note we only care about thresholds for torque applied to the 2nd and 4th joint and the cartesian force in the Z axis. These values will determine/showcase what the robot should do if there is contact.

In option 5 which is our bitwise step implementation, the user can now apply some external force/torque in either direction to the 1st joint and will notice the needle going up and down depending on the direction of torque/force applied to the 1st joint. In the case of contact, you should notice that the threshold for force/torque applied has changed and now more torque/force will be needed to move the needle.

In option 4 which is our impedance control, instead of relying on the 1st joint and reading torque/force input we simply just read the change in joint 1, and depending on the amount that was changed we just adjusted the z value of the end effector accordingly which will cause the robot to move up or down depending if the change it detected is positive or negative. Also, note that there are no contact conditions set so in the case of contact a reflex error may be thrown or the robot might try to power through the contact and attempt to function as normal. This option does produce a smooth motion, but since it uses impedance control, it can be easily manipulated by external forces applied to the end-effector. The error margin that the robot is trying to achieve with impedance is not enough for accuracy, but this option exists as a proof of concept.

Finally to exit the program simply just ctrl+c.

Mohamad Sabagh, Muhammed Saad Qadri

Code Implementation Process

To implement the code for this biopsy program, we started by taking a copy from PS2 code that does the registration and reaching any voxel point in the image space. We had to change some settings in the libfranka desk end-effector to accommodate the needle's weight and length. We modified the code such that it is able to hover above a voxel point and be ready for further instructions.

After that we started to look for similar examples in the libfranka library, and we identified that the 'cartesian_impedence_control' module was very similar to what we are trying to achieve. The most important aspect of this code is the ability of manipulating different joints while the robot is able to maintain the end-effector position with some margins of errors. We modified this code such that it saves the angle of joint 1 and reads any changes on it. The change is then scaled appropertiatly and applies to the target position's Z axis value. This gave us the first proof of concept of using a redundant joint as an input method. The only thing wrong with impedance is that the error margin. While we can redue the stiffness rotationally and translationaly, the robot will start to twitch since it over compensate for every movement. We had to tune the stiffness appropriately, but even with that the error margins where huge and can easily change the end-effector position. Thus reducing accuracy, which is undesired outcome for biopsy procedures.

The next step was to look for alternatives, and found the 'generate_elbow_motion' module to be the next best fit. This module moves the elbow and end-effector position as a function of time. The problem is that during the control loop, the robot is not able to be manipulated by the user, and we couldn't use the same approach as we did for the impedance control to read joint 1 angle. The alternative is that we can read external torque applied to joints and do something with that input. This means that the robot is still not able to move during the control loop, but it can realize when the user is trying to rotate a joint. We used this method to continuously read the torque of joint 1, and when the torque exceeds a threshold, the robot will run the elbow motion control loop for 0.5 seconds. We have to modify the function to fit this short time period. The results is that when the positive torque is being applied, we move the robot's elbow and end-effector position positively for 0.5s and doing the opposite when there is a negative torque.

Now lastly, we had to figure out when the end-effecotr is in contact and how it gives the user feedback proportionally. First we had to go to the libfranka documentation to find more details about the robot.setCollisionBehaviour function. Realizing that it has lower and upper limits for joint torque and forces applied to end-effector, the lower bound is used to determine if there is contact. We tested out lots to tune those parameters, and essentially used multiple readings to identify contact. When the contact is made, the robot will not require higher torque from the user to move, and the distance of movement is reduced as well. This gives more precision and gives the user some feedback when they have made contact with skin or muscles for example.

Mohamad Sabagh, Muhammed Saad Qadri

Description of Testing and Solutions

There were various stages during the production of our project where we conducted various amount of tests:

- Testing our point registration in voxel space to make sure we got to a hovering-over tumor state
    - This was done similarly to PS2, we determined the distance of our tumor in voxel space for all x,y, and z coordinates and used that as input in our programs with some modification to the z coordinate, which is then converted from voxel space to meters and applies the transformation to the robot the get to the hovering state. We tested this many times by giving our program the voxel values we determined and made sure it got to that same hovering state every time.
- Testing the stiffness set in impedance control implementation to avoid the robot's uncontrolled twitching movement with low error margins
    - This is another problem we encountered oftentimes when pushing or moving the robot, we would notice that it would start to twitch and start vibrating. After talking to the TAs, we figured out this was due to increasing the stiffness and dampening values that exceed an error margin in the robot's end-effector. This means that the robot is trying to overcompensate for each movement to keep the end-effector in a small margin of error from it is meant to be and causes the vibrations. We tested this by playing around with the impedance control implementation by pushing the robot around to see if this vibrating action would occur again and adjusting the values again when needed till we got to a stable configuration that could operate with no errors.
- Testing the stability and consistency of our bitwise vs impedance control implementations of our biopsy
    - After making sure that both of our programs were working as they should be we then tested and compared the stability and consistency of each of these implementations. As expected the bitwise implementation which reacts based on torque applied to it was the winner here as the impedance control implementation had to adjust for any movement applied to the end effector and it always had some sort of margin of error to it.
- Testing to avoid acceleration discontinued error
    - This is another error we encountered many times during our project in the bitwise implementation. We were essentially moving too fast for the robot and there needed to be some sort of continuity for it to suitably and safely get to the desired position. How we tested this and eventually got this to work by playing around with the desmos graphing tool and the cosine formula to get a suitable value to increase or decrease the z value each time in a controlled manner.
- Testing to see if the direction of torque applied can influence positive and negative change in Z
    - This was another challenge that we faced during our implementation mainly with dealing with the robot control loop in conjunction with the acceleration discontinuity error. After figuring out a suitable value to add to Z we applied positive torque to the 1st joint and as expected it moved down, then we just simply reversed the value added to Z by subtracting instead and got it to move up when appalling torque/force in the opposite direction. We were able

to test that this function works through our various tests by trying to poke the jello.

- Testing the contact sensing using built-in libfranka methods
    - robot.setCollisionBehaviour method from the libfranka library has a lower and upper torque threshold and a lower and upper force threshold. Reading the documentation, we identified that the lower threshold of torque and force are the parameters used to realize if the robot made contact with external objects without triggering safety reflex shutdown. Reading this information from robot_state allows us to sense the contact of the end-effector with objects and do something different. We had to change the lower thresholds a lot to realize an optimal position where the end-effector is sensitive to small forces, but we got into the issue that the weight of the end-effector was triggering the same threshold as some configurations. The solution was that we had to use a balance to measure the mass of the end-effector precisely and change those values in the end-effector settings of the robot. Also when contact is realized, the robot will look for a higher torque limit to move the end-effector in the biopsy demo and will move less in the Z direction.
- Testing to figure out force feedback of the bitwise module and the use of soft material for contact sensing
    - This is one of the parts of the project where we had the most trouble in our bitwise implementation as the jello was a soft substance that the robot often did not detect any contact with, especially after it had been poked multiple times during implementations and testing. We decided to add a layer above the tumor that could function as skin or muscle. Through this modification, we were finally able to register and detect if any contact had been made above our thresholds set by setCollisionBehaviour by checking if joints 2 and 4 are getting torque above the limits set and checking if cartesian force in the Z axis above the threshold we set. If these conditions are met, then we can safely assume contact with an object. We tested this many times by trying to poke holes in the jello/outer layer and printing details about the force and torque to see if we were getting the right values when in contact.