Project Report

**Paper Review**

This paper [1] proposes a method for implementing a robot framework which is used for tracking the heartbeat motion in high bandwidth. It is validated by the experiment that this testbed can track the reference motion with an RMS error of 0.66mm, which is promising. The project 2 assigned is to help master the mechanism of this framework under the hood by simulating a system that accurately tracks the beating heart once detects the distance of movement, providing a seemingly static and stabilized image for conducting the surgery. However, it is still difficult to put such a robot into real medical use. Reasons are listed:

**1). Model Mismatch:** The paper discusses the utilization of kinematics calibration to reduce kinematics errors. Data is gathered through encoders and motion trackers, requiring a high-precision and robust state estimator. Additionally, the robot model is influenced by mechanical components, where diverse materials exhibit distinct properties (such as gravity and friction coefficients). This variation in system performance is attributed to these mechanical differences.

**2). Stability and Safety:** There isn't extensive coverage of this topic in the paper, but it's crucial for the entire system. When implementing a controller for medical purposes, ensuring safety is imperative. Consider a scenario where a motor executes an unintended action during surgery, potentially causing irreparable damage to the patient's heart. The importance of safety measures cannot be overstated in such critical contexts. We may not always create a controller with mathematically proven safety assurances, but it's essential to establish certain constraints for the robot system during its operations. However, this aspect is not addressed in the paper.

**3). Realtime Efficiency:** I think this is the very part that hinders the system from being put into practice. The robot system is split into different modules where a sensor detects the heart motion and sends collected data back to controller for execution. Normally we suppose it uses ROS for communication. However there exists delays and data loss during message transmission. The delay could be milliseconds and data loss could be ones or tens, but the impact caused to tracking the heartbeat would be unpredictable because heart is beating in high bandwidth.

**4). System Reliability:** In real-time systems, it is essential to incorporate a design where, if a controller fails, the system can sustain its operational status for subsequent tasks. For instance, if a motor experiences a malfunction or overheating during a process, an alternative motor should promptly be activated and substituted to ensure the continuous execution of tasks, thereby enhancing the system's reliability.

To make further improvements, we can take several measures accordingly. For 1), the difference is typically caused by data. We could use ML and adjust the mismatch by training a robust model; For 2). As said above, we are to make restrictions for the forces/torques applied on the controllers to further constrain the action domain of the end-effector; For 3). Time delay is normally composed of two parts: software delay and network delay. For software delay, it can be optimized by removing the unnecessary threads and processes. For network delay and packet loss, approaches from TCP/IP layers especially designed for sensor network like [3], [4] should be considered for practical use. For 4), a smooth switch approach between action controller and backup controller like [2] can be applied under real plant.

**Project Overview**

Doing Project2 really helps me get to know how the robot system works in a low-level design flow. For the hardware part, the used tools include Arduino master controller, servo motor, ultrasonic module, breadboard, green/red LED, resistors, and numerous cables.

A circuit board with wires

Description automatically generated

For the software part, a GUI using MATLAB App designer is shown below. Users will feel free to load/save collected data and edit transmit pins on it accordingly. A figure of heartbeat will be plot in real time in the middle of GUI once the system is running.

A graph with a white background

Description automatically generated

**Initialization**

To initialize the system, click ‘Initialize’ button. Then the system will automatically detect the connect serial port and load Arduino within its ultrasonic and servo motor library. Besides, it will also initialize ROS2 node as ‘/publisher’ and get ready for publishing the data of type ‘std\_msgs/String’ to the topic ‘/chatter’; Lastly, it would initialize some variables which is designed for following work. If there’s any errors occurring during initialization, an exception will be thrown for debug.

**Play Button**

This is the button to get the system work after initialization. Once clicked and the ultrasonic module detects movement, LED status turns to green, and the system will collect the raw data and filter the noise using a median filter where the windows size is adjustable. The collected raw data and filtered data will be plot in real time as shown below. And the filtered data will be sent in two ways: one is to be encapsulated as a ROS message to the topic by the node; another is being sent as action command for motor to execute. (In reality, the data is sent through ROS to another ROS node – receiver. Motor angles are being calculated and action commands will then be generated and sent for the low-level motor to execute. However, in this project an Arduino port cannot be occupied by two nodes in MATLAB at the same time, because of which we send motor actions here to simulate the real robot controller command for simplicity.) The Gauge represents the ratio of motor rotations: r = rotating angle/max angle.

A screen shot of a graph

Description automatically generated

**Load/Save Button**

The Load Button is for loading the previously saved data while the Save Button is used for storing collected data. The load/save path is predefined in a variable called ‘DataPath’ in the app properties. Please note that once the data is loaded and ‘Play’ button is clicked, the system will continue collecting data following the last time in the stored data rather than collect data from the very beginning of the X/Y axis, which is reasonable. (See below picture, once data is loaded, system will begin collection from the place marked by red pen.)

A screen shot of a graph

Description automatically generated

**Reset Button**

This button is to reset all the variables and data to their initial status and values. In the meanwhile, the plotted figure will be cleared out and the LED light is set to be black (turned off). The servo motor will move to its initial angle and the motor gauge will also be set to zero. But the system won’t have to get initialized again after resetting. Feel free to click the ‘Play’ Button to begin collecting data once ready.

**ROS Node**

As said before, a publisher node will be initialized during the initialization process, and it will keep publishing message to the topic ‘/chatter’ whenever a valid distance information is collected from the ultrasonic module (Note that this distance information is treated as the difference between the actual distance and initial/reference distance). In the meanwhile, we start another ROS node naming ‘subscriber’ in the MATLAB command line to receive the message and process it using a callback function named ‘ros2callback’. Inside this callback function, A 6Dof PUMA560 robot is leveraged as the testbed for this simulation. The subscriber node reads the distance data and uses inverse kinematics to calculate the corresponding joint angles for moving the robot end effector to the right position (Also note that here the desired movement for robot is calculated along the X-axis for simplicity). Once you’re running the two nodes and valid data is got transmitted, you will find that PUMA560 robot will move forward or backward within the same distance to track the beating heart motion.

A screenshot of a computer

Description automatically generated

**Video Link**

YouTube Channel: <https://www.youtube.com/watch?v=owcGzDJOgxw>

**Experiment**

As mentioned above, the data is collected during sampling and the figure is plotted in real time which contains both the original raw data and the filtered data using a median filter.

A graph with blue lines and red lines

Description automatically generated

Here I’ve collected three different frequencies for comparison:

1). From the figure below, the frequency of beating heart would be around 0.6 – 0.7 Hz. The heartbeat rate would be 36 – 42 BPM.

A graph with blue and orange lines

Description automatically generated A graph with numbers and symbols

Description automatically generated

2). For this figure, the frequency of heartbeat is approximately 0.4 – 0.5 Hz; The corresponding heartbeat rate would be 24 – 30 BPM.

A graph with blue and red lines

Description automatically generated A graph with numbers and symbols

Description automatically generated

3). By this collection, the frequency of heartbeat is approximately 0.7 – 0.85 Hz; The related heartbeat rate is 42 – 51 BPM.

A graph with blue and orange lines

Description automatically generated A graph with numbers and symbols

Description automatically generated

For the time delay. I don’t have a good approach to calculate the ROS transmission period because the collection node and the execution node are separated, where the processing period cannot be directly obtained using the tic/toc function in MATLAB. My thinking is that we could set a standard time for these two nodes, and the collection node sends out the current time to a time topic while it is sending out the ROS message to the execution node. The execution node will also record the timestamp when it receives the distance information and broadcast the recorded timestamp to the time topic. A third node will subscribe to the time topic and compare the difference between two timestamps to get the actual time delay for ROS transmissions. I am not sure whether this is feasible in the real time system but it’s just a thinking about the approach coming out from my mind…

**Reference**

[1] O. Ersoy, M. C. Yildirim, A. Ahmad, O. D. Yirmibesoglu, N. Koroglu and O. Bebek, "Design and Kinematics of a 5-DOF Parallel Robot for Beating Heart Surgery," 2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM), Toyonaka, Japan, 2019, pp. 274-279, doi: 10.1109/ICARM.2019.8833632.

[2] Seto, Danbing and Lui Raymond Sha. “An Engineering Method for Safety Region Development.” (1999).

[3] Dong, Mianxiong, et al. "Joint optimization of lifetime and transport delay under reliability constraint wireless sensor networks." IEEE Transactions on Parallel and Distributed Systems 27.1 (2015): 225-236.

[4] Liu, Yuxin, et al. "QTSAC: An energy-efficient MAC protocol for delay minimization in wireless sensor networks." IEEE Access 6 (2018): 8273-8291.