EN 601.656 Computer Integrated Surgery II Final Report

IMPROVING THE TRANSPARENCY OF THE GALEN ROBOT

By

Vishnu Kolal

(Project 29 – 9th May 2022)

Mentors: Dr. Russell H. Taylor, Dr. Adnan Munawar

I. ABSTRACT

The Galen Surgical Robot is a prototype delta platform based robotic manipulator. It is designed to be cooperatively operated in a hand-over-hand control mode by an experienced surgeon and intended for use in Otolaryngology head and neck operations. The Galen Robot when operated in hand-over-hand control mode should allow the surgeon to manipulate the tool in a fluid and transparent manner with high responsiveness. However, the robot in its current state leaves much to be desired in terms of transparency. This report presents various steps taken to improve the responsiveness, fluidity and transparency of the Galen robot. Multiple approaches were used to try and achieve this goal, including admittance gain tuning, control loop timing correction, sensor data filtering and adaptive gains. As a result of these improvements, the robot's transparency improved and it became more responsive to operator input.

II. INTRODUCTION

The Galen Robot (Mk-2) is a prototype Delta Platform based robotic manipulator intended for surgical use. It comes equipped with a 6-axis F/T sensor mounted between the robot and the end effector. The sensor provides vital information which is used to control the robot in a hand-over-hand control mode where a surgeon and the robot, both, hold the tool together. The surgeon controls the tool path directly while the robot provides a steady platform by eliminating hand tremors and enforcing hard or soft limits. The tool is typically a surgical drill intended for cutting bone. The robot has great potential to improve the safety and efficacy of certain surgical procedures including Laryngeal surgery, Stapes Surgery, Mastoidectomy, etc. However, the robot is a prototype and needs significant improvement before it is ready for use in a clinical setting.

Figure 1: Galen Robot Mk2. Image curtesy www.galenrobotics.com

III. PROBLEM STATEMENT

The goal of this project, as stated in the title, is to improve the transparency of the Galen hand-over-hand surgical robot i.e. to make the tool feel ‘weightless’ in the hands of the operator. Currently, the feel of the tool while used in the hand-over-hand control mode leaves much to be desired. This project aims to improve this ‘feel’ and make the robot more responsive and transparent to operator force input.

IV. SIGNIFICANCE

The Galen hand-over-hand surgical robot has potentially limitless applications such as eliminating hand tremors, imposing virtual fixtures that prevent the operator from damaging sensitive organs, drilling out of pockets in bone structures with near prefect precision, automated guided biopsies and maybe even fully automated surgery in the not-so-distant future.

There are also other ongoing projects in LCSR that aim to further develop the Galen robot such as the force sensing drill project, the force sensing forceps project, virtual reality guided skull base surgery, etc.

All of these potential applications and projects would benefit greatly from an improvement in the transparency of the robot control.

V. TECHNICAL APPROACH

In summary, the approach to improving the transparency and responsiveness of the Galen Surgical Robot was to make changes or add improvements to the admittance control algorithm and test the robot in a repeatable manner to verify if the changes resulted in an improved transparency or a deteriorated transparency. This project was a controls systems tuning project in essence and was thus fairly open ended in terms of possible approaches.

However, before any changes could be made, an objective method to determine whether a change to the admittance control algorithm resulted in an improvement in transparency or not. The magnitude of force exerted on the wrist F/T sensor was chosen as a metric to determine improvement in transparency. This is due to its correlation as an indicator for transparency. If the robot in hand over hand control mode is more fluid and responsive, the operator will have to exert lesser force on the wrist F/T sensor.

Having established verification metrics, the experimental setup and the various approaches tested for improvement in transparency are described below.

1. Experimental Setup

A simple experimental setup was used to test the robot in each case. The experimental setup consisted of a sheet of paper with a 10x10cm square drawn on it, placed beneath the robot. A laryngeal forceps tool was mounted on the robot and the robot was used in hand-over-hand control mode to trace the square path on the paper. The robot was moved at as constant a velocity as possible during this motion. The motion was repeated three times and the forces on the wrist sensor were recorded. Iterations with a lower mean force magnitude were deemed desirable.

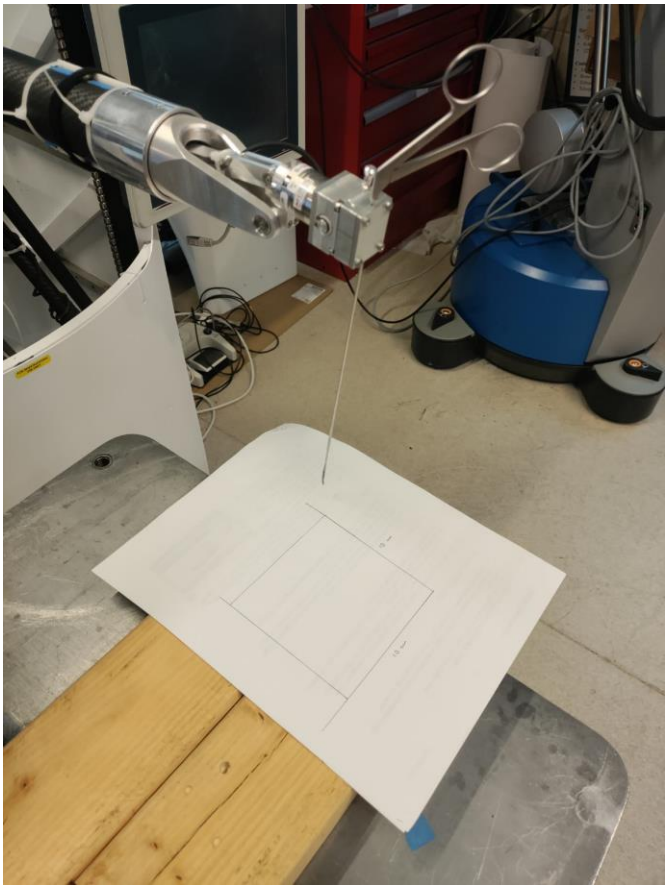


Figure 2: Experimental setup for testing transparency.

2. Control Loop Timing

The robot uses two different control loops running at the same time to function properly. The low-level Galil controller is used to control the motion of the individual motors on the robot. It receives velocity goal commands from the mid-level controller and uses PID control to appropriately set the motor currents. The mid-level controller runs an admittance control algorithm to calculate the velocity goals depending on the commanded force input at the wrist F/T sensor. If the low-level Galil controller does not receive these velocity commands at regular intervals, it may result in small instabilities and reduced responsiveness.

There are multiple reasons why the command velocity intervals may vary but the primary reason is variability in the amount of time required to calculate the velocity goals. The mid-level controller uses a numerical solver which may take varying amounts of time to converge at a solution, especially if more complex constraints such as virtual fixtures are implemented.

In order to rectify this, the timing of the commands being sent to the Galil controller was changed. The algorithm originally sent commands to the Galil as soon as it completed the calculations for the goal velocities. This was changed in such a way that the goal velocities from the previous iteration are sent at the beginning of each control loop and the velocities for the next iteration are then calculated and stored. The controller then sleeps for the remaining time.

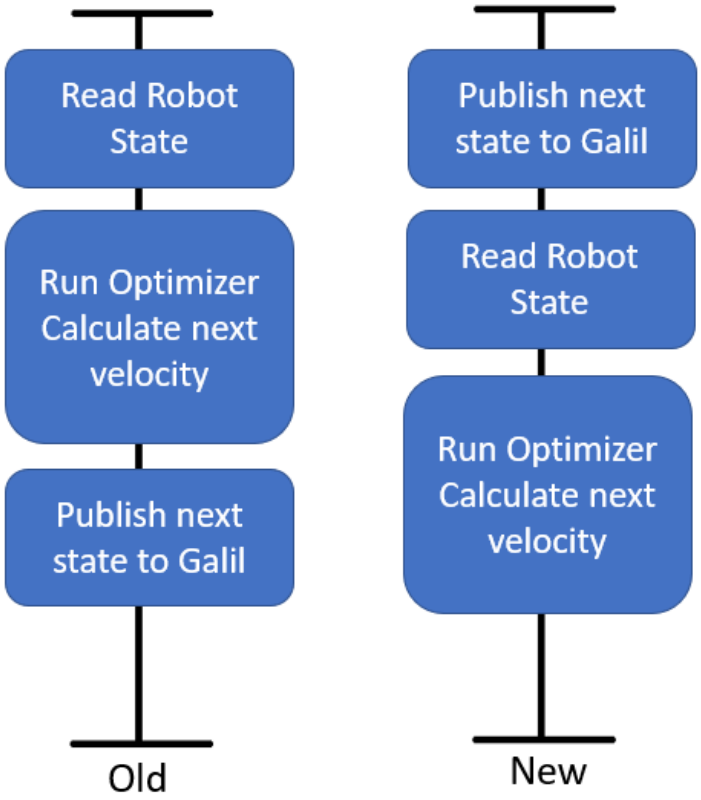


Figure 3: Block diagram of old and updated control loop.

3. Adaptive Gains

The Galen Surgical Robot uses an admittance control algorithm to operate in hand-over-hand control mode. Since the robot has a wide range of potential applications, a 'one size fits all' approach to the admittance gains may not be the best option. In most cases, two types of robot motions are commanded by the operators, namely, precise motions and transient motions. Precise motions are common when the operator is performing a surgical task in a tight space such as drilling out a pocket in a bone or attempting to precisely grip some tissue. In these cases, the commanded motions are very small and thus, a lower gain is suitable. Conversely transient motions are when the operator brings the robot into position for a precise task or moves the robot away after completion of a precise task. These motions are much larger and do not require precision but instead require speed. Therefore, higher admittance gains are suitable for transient motions.

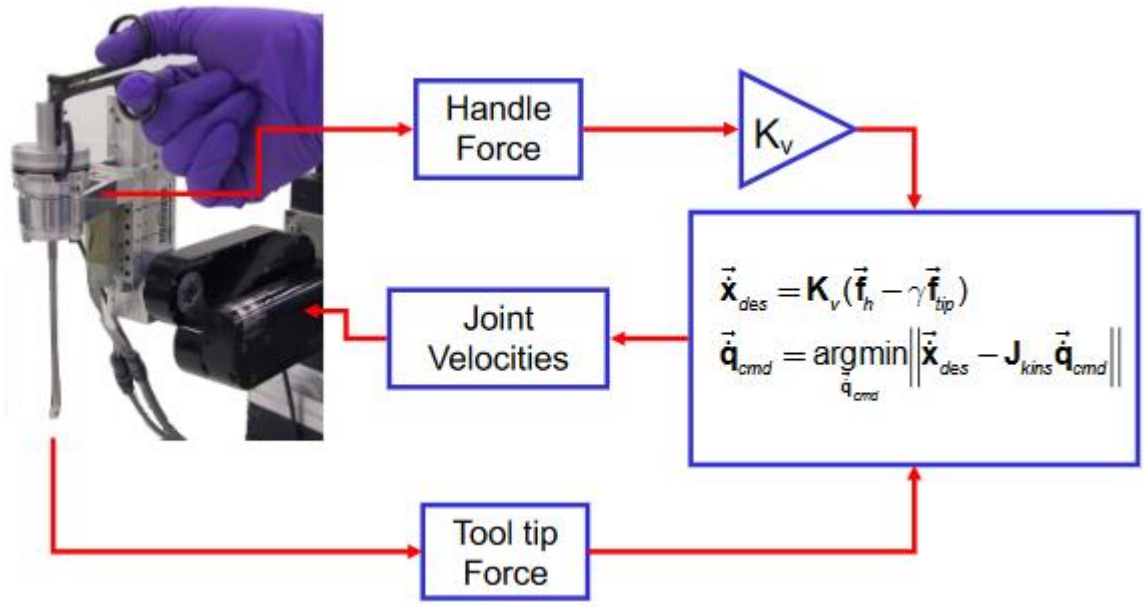


Figure 4: Admittance control block diagram. Source: CIS-1 lecture notes, R.H. Taylor

To tackle this problem, an adaptive gains implementation was used to dynamically increase or decrease the gain based on the operator commands. An adaptive gain factor was introduced in the control loop and varied in accordance with the force input on the wrist force sensor.

The adaptive gain was implemented in such a way that precise and transient motions are detected based on the magnitude and duration of force. If the magnitude of force applied on the wrist F/T sensor is more than a certain threshold for a certain duration, the gains are slowly ramped up to a maximum limit. Conversely, if the magnitude of force on the wrist sensor is below the threshold or the foot pedal is not completely depressed, the gains are transitioned back to their base value. A number of different parameters were also introduced to tune this implementation of adaptive gains. The descriptions are as follows:

i. Force Threshold: The threshold force above which adaptive gains are triggered.
ii. Max Limit: Maximum allowable adaptive gain.
iii. Min Limit: Minimum allowable adaptive gain.
iv. Time Threshold: The time period in between when the force threshold is triggered and the adaptive gain starts to ramp up
v. Ramp Up Time: Time taken for adaptive gains to ramp up to maximum limit.
vi. Ramp Down Time: Time taken for adaptive gains to return to the base level from the maximum limit.

4. Sensor Data Filtering

When the Galen Robot is operated in hand-over-hand control mode, it provides input to the admittance controller in the form of force/torque from the wrist F/T sensor. The data measured by this sensor directly influences the motion of the robot and any noise or inaccuracies present in these measurements can affect the responsiveness and stability of the robot.

At higher admittance gains, the robot end effector is subjected to accelerations and decelerations of higher magnitudes. In these situations, the force sensor is likely to pick up forces and torques resulting from the inertia of the tool. This could lead to undesirable effects.

To prevent these undesirable effects, a simple low-pass FIR filter was implemented. The filter has a buffer size of 20 and uses simple triangular weighting.

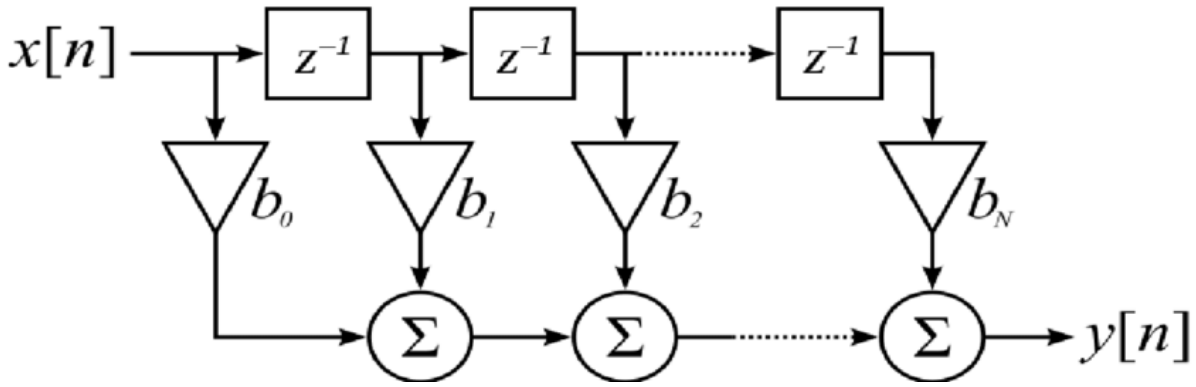


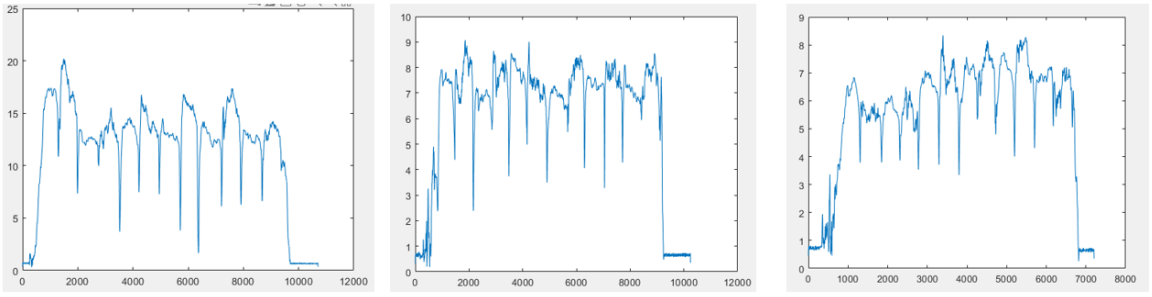
Figure 5: Block diagram of low pass FIR filter. Source: www.researchgate.net

VI. RESULTS:

Each of the various tasks completed over the course of this project improved the transparency and responsiveness of the Galen Robot in an incremental manner.

The primary datasets to demonstrate the implementation of the technical approach described above are force sensor data, robot end effector position and velocity data, control loop refresh rate, and user feedback. Although the trials and analytics of data collected from the robot after the changes were implemented are yet to be completed, some of the preliminary results follow.

The robot when tested in the experimental setup with varying gains demonstrated some reduction in wrist force sensor magnitude with increasing gains. As shown in the below image, although the forces appear to show a downward trend of force magnitude vs gain, the sensor in the experimental setup, as operated by a user, is subject to a significant amount of noise and inconsistencies. For this reason, the experimental setup was phased out as a metric and only used for reference. The preferred metric for the remainder of the project was user feedback from people with experience on the robot..



Mean:13.36 Gain=1	Mean:7.13 Gain=2(default)	Mean:6.27 Gain=4

Figure 6: Plots of Sensor Force (N) vs time with varying gains.

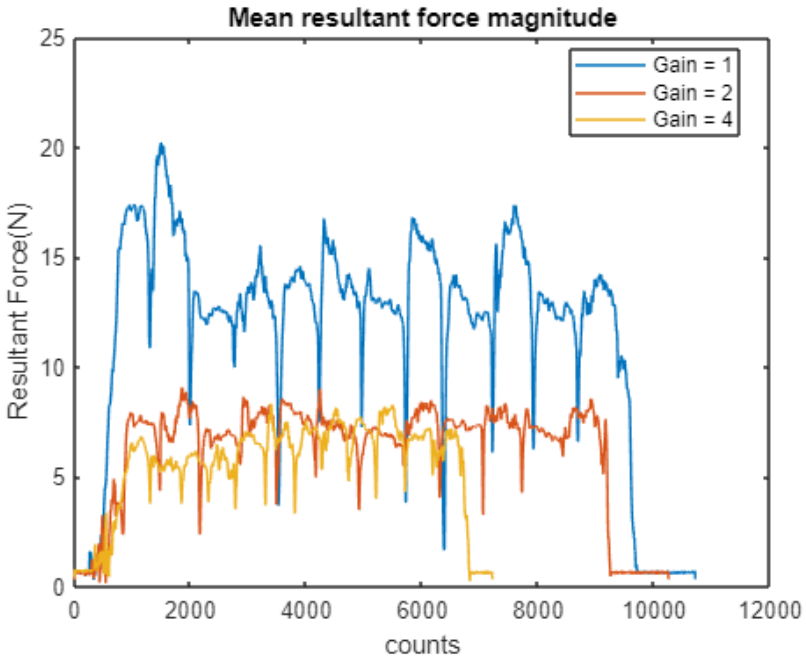


Figure 7: Comparison of Sensor Forces at different Gains. At higher gains, plots are similar in magnitude.

The changes made to the control loop timing have a subtle effect on the transparency and responsiveness of the robot. In most use cases, the mid-level admittance control loop does in fact, send commands to the low-level Galil controller at regular intervals. However, when there are heavy constraints on the admittance controller and it is forced to dedicate more time to converge at a solution, the timing may vary. Nevertheless, as shown in the below image, the mid-level controller is able to maintain a steady 198Hz and sends commands to the Galil every 5.05ms

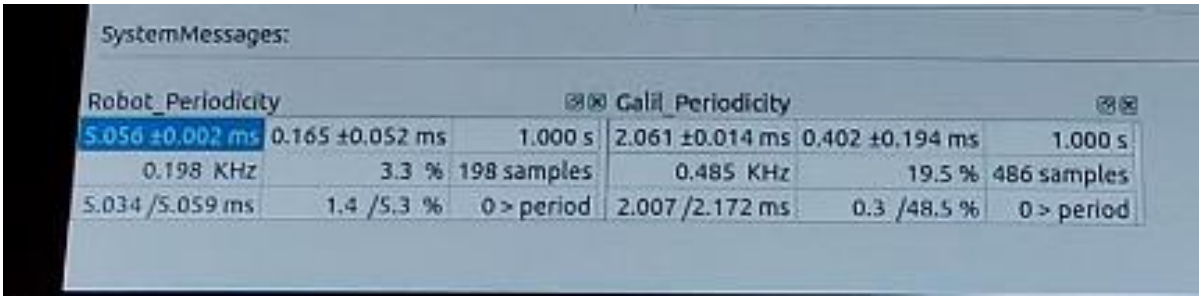


Figure 8: Periodicity and performance metrics displayed on the Galen Robot GUI

The most significant improvement to transparency and responsiveness came with the addition of adaptive gains. This implementation allowed for separate tuning of gains for the two common surgical scenarios instead of a single gain. This allowed for precision control for small motions and high velocity for large motions. The approximate timing of the adaptive gains, threshold and scale factor is illustrated below:

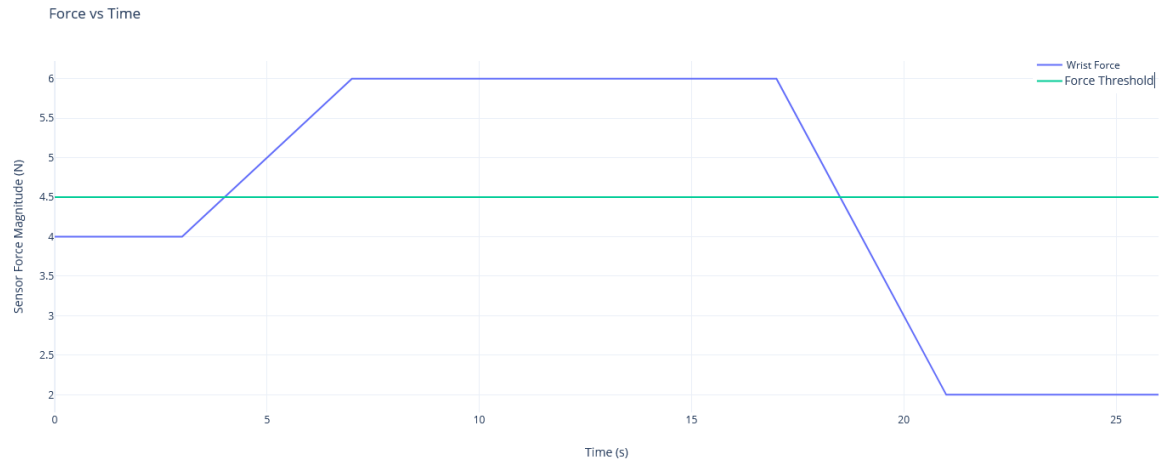


Figure 9: Illustration of Sensor Forces vs Time

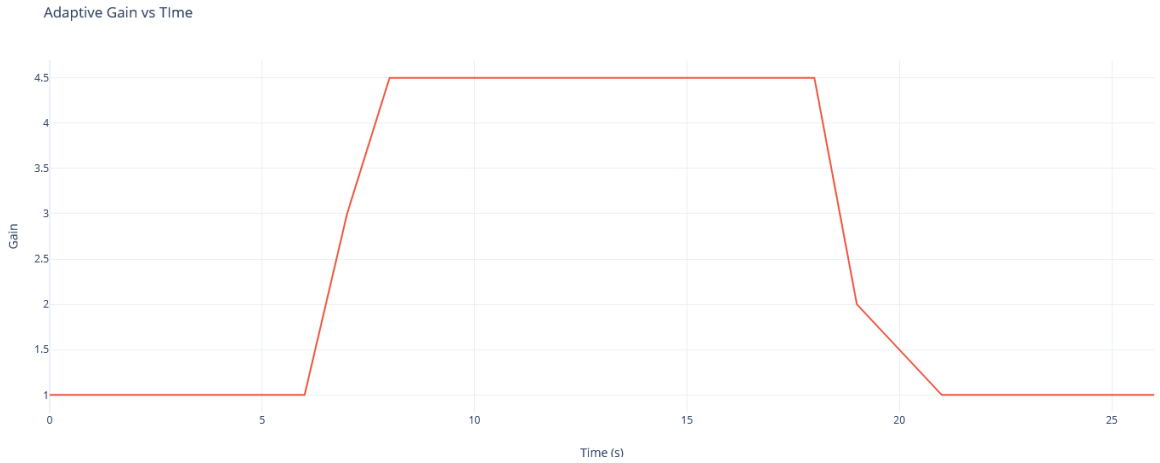


Figure 10: Illustration of rise and fall of Adaptive Gain vs Time

These characteristics were experimentally verified using the same experimental setup described above. During the experiment, force data, velocity data, and adaptive gain was recorded. The same experiment was repeated with and without adaptive gains. A comparison of the robot running with and without adaptive gains follows:

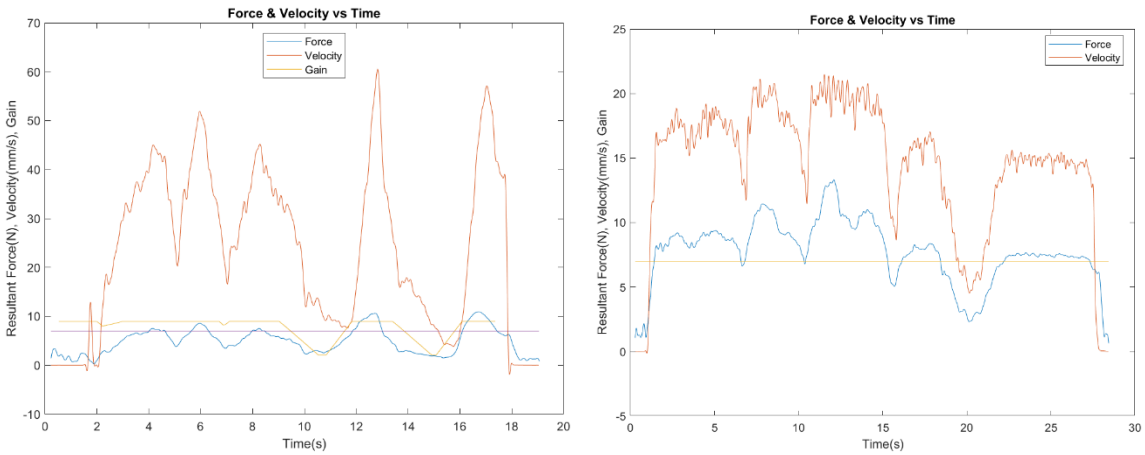


Figure 11: Full experiment with adaptive gains (left) vs without (right)

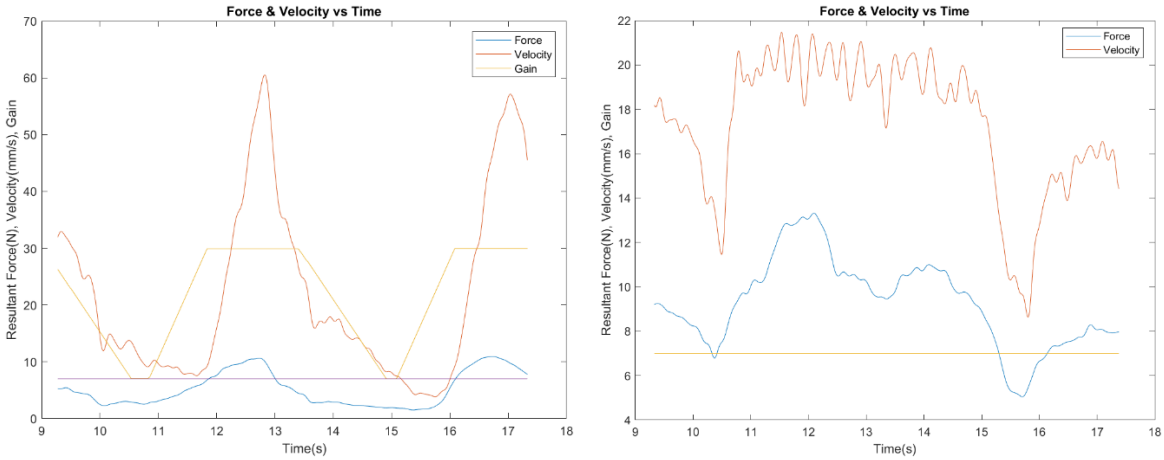


Figure 12: Part of experiment with adaptive gains (left) vs without (right)

The impact of sensor data filtering on the transparency and responsiveness of the Robot is currently unknown and the analysis of this data is ongoing.

VII. MANAGEMENT SUMMARY

1. Roles: Since I was the sole member of this project, I did all the coding and testing. I would also like to acknowledge the assistance provided to me by Tommy Liang. Especially with familiarizing me with the Galen Robot's Source Code and assistance with the GUI.

2. The original deliverables for this project are as follows:
   - Minimum:
     a) Code cleanup & corrected control loop timing
     b) Tuned admittance gain at maximum stable limit
   - Expected:
     a) Mean sensor forces reduced by 50%
     b) Experimental data and operator testing result analytics
   - Maximum:
     a) Mean sensor forces reduced by 75%
     b) Implemented adaptive gains control loop
     c) Control loop with wrist force and tool-tissue force

However, in the first month after starting this project I realized that some of the deliverables were not feasible due to the nature of the robot. Especially the deliverables pertaining to the mean sensor forces. The forces exerted on the wrist sensor of the robot tend to vary from application to application but even at very high admittance gains, the forces do not reduce below 80%. Although the measurement of the mean sensor forces provides a good indicator for transparency and responsiveness of the robot, and were extensively used, these deliverables would not be met. Upon consultation with Prof Taylor, these deliverables were scrapped in favour of more realistic ones.

A second major hurdle in this project was the Galen Robot itself. The robot faced some hardware issues and was intermittently operable for approximately a month and then completely inoperable

for three more weeks. Since my project depended on the robot heavily for testing, this was a major setback. During this time, I was able to continue writing code but unable to test on the robot. The robot was eventually replaced I was able to debug my code and gather results.

I was also able to complete a significant amount of my updated deliverables that are as follows:

- Minimum:
  a) Code with corrected control loop timing - COMPLETE
  b) Documentation describing changes - COMPLETE
- Expected:
  a) Tuned admittance gain at maximum stable limit - COMPLETE
  b) Experimental data and operator testing result analytics - ONGOING
- Maximum:
  a) Code with adaptive gains control loop implemented – COMPLETE

3. Next Steps:

The immediate next steps for the scope of this project is to further tune the parameters of the admittance controller and the adaptive gains parameters based on surgeon feedback. Due to delays with the robot, these trials and analytics could not be completed on time. This is a necessary step and work on this will continue in the summer.

One of the long term planned tasks that I was unable to implement was a dynamic model of the robot. The implementation of a dynamic model for the robot and plant model for admittance control and PID control is an essential step in improving the transparency and responsiveness of the robot. This task was out of the scope for a one semester long project, therefore, an attempt to implement this will be made in the future.

4. Learning Experience:

As a mechatronics engineer, I briefly studied control theory in my undergraduate years, however, I had not gotten a chance to work on a large scale project such as this and my experience in controls engineering was limited to PID tuning. I was thrilled at the opportunity to work on the admittance controller of a complex medical robotics system like the Galen Robot and I was prepared to face any challenges head on.

This project helped add context to what I learnt in CIS-1 and I learnt a lot about admittance controllers in the process of tuning the gains. The experience I gained while debugging my code is also invaluable to me and helped me become a better C++ programmer.

A personal goal for me during this project was to familiarize myself with the working of the Galen Robot in preparation for a potential internship at Galen Robotics. I now feel that I have achieved this goal.

VIII. TECHNICAL APPENDIX

The culmination of this project was implemented in a new force control behavior titled ‘ForceControlBehaviourAdaptive’ in the ‘ForceControlBehaviours’ Class. This adaptive mode is accessible to the user through the Galen Robot GUI and is titled ‘FTVELOCITY_ADAPTIVE’. This mode of operation can be selected instead of the customary ‘FTVELOCTY’ option.

```
ForceControlBehavior::ForceControlBehavior(robotTask *robot, std::shared_ptr<robotModel> model, std::shared_ptr<IGalenLogger> logger)
BehaviorBase("ForceControlBehavior", model, logger),
StateTable(5000, "StateTable"),
RobotTask(robot),
m_pLogger(logger)
{
  AddMode(Mode::BehaviorManagerStrings::LinearForceControl);  //mode 1.
  AddMode(Mode::BehaviorManagerStrings::LinearForceControl_SIM); //mode 2.
  AddMode(Mode::BehaviorManagerStrings::LinearForceControl_Adaptive); //mode 3.
```

To implement the corrected control loop timing, two new functions were added to the 'RobotTask' class. The function 'SavePrevJointVelSet' can be used to store current joint velocity goals in memory after they have been calculated by the optimizer. The function 'GetPrevJointVelSet' can be used to retrieve the joint velocity goal calculated in the previous timestep.

```
//store the velocity goal after optimizer calculates it (CIS2 project Vishnu Kolal)
void robotTask::SavePrevJointVelSet(const mtsDoubleVec& prevVelSet)
{
    prevJointVelSet = prevVelSet;
}

//retrieve the velocity goal to send to galil (CIS2 project Vishnu Kolal)
mtsDoubleVec robotTask::GetPrevJointVelSet() const
{
    mtsDoubleVec retVec;
    retVec.SetSize(5);
    retVec.Zeros();

    retVec = prevJointVelSet;

    return retVec;
}
```

Implementation of adaptive gains:

```
void robotTask::UpdateJointGains(void)
{
    if(PedalLR[0] > 0.0)
    {
        // p * mf * (maxVel * dtOpt)/maxFT
        gains[0] = PedalLR[0] * SliderGains_R * scaleFactor[0];
        gains[1] = PedalLR[0] * SliderGains_R * scaleFactor[1];
        gains[2] = PedalLR[0] * SliderGains_T * scaleFactor[2];  //invert to correspond to corrected FT
    }
    else
    {
        gains[0] = gains[1] = gains[2] = 0;
    }

    mtsDouble nextAdaptiveFactor;
    if(PedalLR[0] > 0.9 && forceAtHandle.XYZ().Norm() > adaStartForce)
    {
            if(adaElapsedCount * StateTable.Period < adaStartPeriod)
            {
                adaElapsedCount++;
            }
            else if(adaElapsedCount * StateTable.Period >= adaStartPeriod)
            {
                nextAdaptiveFactor = adaptiveFactor + ( (adaptiveMax - adaptiveMin) / ( adaRampUpPeriod/StateTable.Period ) );                          //ramp up
                if(nextAdaptiveFactor * scaleFactor[2] <= adaMaxGain && nextAdaptiveFactor <= adaptiveMax) { adaptiveFactor = nextAdaptiveFactor;}
            }
    }
    else
    {
        if(adaptiveFactor > adaptiveMin)
        {
            adaptiveFactor = adaptiveFactor - ( (adaptiveMax - adaptiveMin) / ( adaRampDownPeriod/StateTable.Period ) );                          //ramp down
        }
        else if(adaptiveFactor <= adaptiveMin)
        {
            adaptiveFactor = adaptiveMin;
        }
        adaElapsedCount = 0;
    }
```

The adaptive gains variable ‘adaptiveFactor’ was added to the ‘UpdateJointGains’ method in the class ‘RobotTask’. This variable is kept updated at all times but is only being used by the adaptive force control behavior.

The variables being used for the adaptive gains are initialized in the constructor for the ‘RobotTask’ class and can be edited here.

```
//variables for adaptive gains (CIS-2 project -Vishnu Kolal)
adaptiveFactor = 1;         //adaptive gain (should be 1)
adaptiveMin = 0.6;            //min adaptive gain
adaptiveMax = 5.8;          //max adaptive gain
adaElapsedCount = 0;        //counter for duration (should be 0)

adaStartForce = 4.20;       //Force above which duration starts (N)
adaStartPeriod = 2.80;     //duration after which adaptive gain kicks in (s)
adaRampUpPeriod = 2.1;      //duration for the adaptive gain to reach its max limit (s)
adaRampDownPeriod = 0.40;    //duration for the adaptive gain to come back to normal (s)
adaMaxGain = 15;            //max scalefactor*adaptiveFactor allowed in adaptive mode
adaTemp = 0;                //ignore
```

Implementation of sensor data filtering:

```
if(isRawFTDataOkay && isPostAlignmentFTDataOkay)
{
    //Applying Frame Transformations
    forceAtHandle = robotModel::TransformForceToOutputFrame(forceAtEE, toolHandleFrame);
    model->ApplyThresholdToFTVector(forceAtHandle, ForceThreshold.Data, ForceSensor::torque_threshold_Nm);
    for(size_t i=filterBuffer; i>0; i--)
    {
        forceAtHandleFilt[i] = forceAtHandleFilt[i-1];
    }
    forceAtHandleFilt[0] = forceAtHandle;
    mtsDouble forceWeight = 1;
    forceAtHandleSmooth.Zeros();
    for(size_t i=0; i<filterBuffer; i++)
    {
        forceWeight = forceWeight/2;
        for(size_t j=0; j<6; j++)
        {
            forceAtHandleSmooth[j] = forceAtHandleSmooth[j] + forceWeight*forceAtHandleFilt[i][j];
        }
    }
```

This algorithm was also implemented directly in the ‘UpdateFTData’ method of the ‘RobotTask’ class and is called in every iteration. The implementation of this sensor data filtering does not affect other modes of operation of the robot because the filtered data is stored in a separate variable called ‘forceAtHandleSmooth’ which is only being used by ‘ForceControlBehaviourAdaptive’. This filtered force is however, available to all methods.

IX. REFERENCES

[1] K. Olds, *Robotic Assistant Systems for Otolaryngology-Head and Neck Surgery,* PhD thesis in Biomedical Engineering, Johns Hopkins University, Baltimore, March 2015.

[2] K. C. Olds, "Global Indices for Kinematic and Force Transmission Performance in Parallel Robots", *IEEE Transactions on Robotics,* vol. 31- 2, pp. 494-500, April, 2015.

[3] A. L. Feng, C. R. Razavi, P. Lakshminarayanan, Z. Ashai, K. Olds, M. Balicki, Z. Gooi, A. T. Day, R. H. Taylor, and J. D. Richmon, "The robotic ENT microsurgery system: A novel robotic platform for microvascular surgery", *The Laryngoscope,* 127, pp. 2495-2500, November, 2017.

[4] Optimal Design of FIR and IIR Filters using some Evolutionary Algorithms - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Block-Diagram-of-a-FIR-Filter_fig3_261213826 [accessed 10 May, 2022]