Anatomical virtual fixture assisted mastoidectomy

I. Basic Info (->Summary)

- **Student**: Yiping Zheng
- **Mentor**: Max Li, Dr. Taylor
- **Group Size**: 1
- **Group No.** 7
- **Skills Requirement**: CISST-SAW library, Slicer, ROS, C++, motion planning, nonlinear control
- **Goal**: Create a motion planning demo for Galen robot to perform robot-assisted mastoidectomy task

II. Background, Specific Aims, and Significance

Most of surgeries require highly precise manipulation from surgeons which can be very time-and-energy consuming. For example, Mastoidecotmy is a delberate surgical procedure which is important to the treatment of diseases such as cochlear implant, acoustic neuroma etc. Surgeons have to mechanically drill a hole on patient's skull and all the way down to the meningeal, meanwhile carefully avoiding sensitive anatomy structure such as facial nurve, sigmoid sinus, and arteries. The drilling process is a challenging process to surgeons which often lasts 8 hours.

Surgical robots such as da Vinci Surgical System can mitigate the challenges by extending human capabilities. However, mainly because of the precision requirement of mastoidectomy is very high, so far there hasn't been any application or attempt of robot assisted mastoidectomy to our best knowledge.

Virtual fixture is software motion constraints, can further reduce the operational difficulties by allowing the surgeon and robot to work together to complete the surgical task with improved stability, reliability and precision. By tracking the relative position of the surgical tool with regard to patients body, it can stop the surgeon from making sudden motions and accidentally damaging critical anatomies and have the potentiality to make the robot assisted mastoidectomy possible. However due to the difficulty in finding a good geometry representation with respect to the geometric complexity of anatomical tissues, the application of virtual fixture is limited. So far no one has attempted to apply it to very complex anatomies such as temporal bone.

Recently, a new virtual fixture generation algorithm was proposed which fits the scenario of mastoidectomy well. By obtaining the 3D data of patient's skull, either pre-operatively via CT scan or intra-operatively via 3D ultrasound, it can generate virtual fixtures online from polygon mesh representations of complex anatomical structures and provide dynamic constraint formulation for the planning and algorithm. The algorithm has been testified through validation and runtime experiments.

In this project, I'm going to integrate this anatomical virtual fixture generation algorithm with Galen surgical robot and perform a demo of robot-assisted mastoidectomy, which may be very useful to avoid touching patient's critical anatomy structure and mitigate surgeons' tension in the procedure.

III. Technical Approach

Since our goal is to follow the new algorithm proposed in [1], create a demo based on it and possibly improve its performance. So in Part 3.1~3.3 of the approach section, the approach of [1] will be presented again and followed with some algorithmic improvement of my own, which is in Part 3.4~3.9.

3.1 Constraint Optimization

Constraint optimization approaches are well-established methods to implement virtual fixtures. In this project, we formulate robot kinematic motion control as a quadratic optimization problem with linear constraints. Objective function solved for the desired motion:

$$\arg\min_{\Delta \boldsymbol{q}} \|\Delta \boldsymbol{x} - \Delta \boldsymbol{x_d}\|_2$$

$$subject\ to \quad \boldsymbol{A}\Delta x \geq \boldsymbol{b}, \ \ \Delta \boldsymbol{x} = \boldsymbol{J}\Delta \boldsymbol{q}$$

where $\Delta \boldsymbol{x}$ and $\Delta \boldsymbol{x_d}$ are the computed and desired incremental Cartesian positions, and $\Delta \boldsymbol{q}$ is the incremental joint position. $\boldsymbol{J}$ is the Jacobian matrix relating the joint space to the Cartesian space. $\boldsymbol{A}$ and $\boldsymbol{b}$ are matrix and vectors necessary to describe the linear constraints.

In a telemanipulated control environment, $\Delta \boldsymbol{x}$ in the objective function is computed from the motion of the master manipulator. Additional objectives may be added to express additional desired behaviors. Otherwise, in cooperative control setting, $\Delta \boldsymbol{x}$ is computed from the force sensor input.

Inequality constraints can be used to impose motion constraints. For example, a virtual forbidden wall for tool tip x can be defined by a hyper-plane with normal n and point p , i.e. tool tip can only move on the positive side of the hyperplane as shown in **Fig. 1**. This can be achieved by forcing the signed distance d_t from the tool tip to plane at any time t to be positive, i.e.,

$$\begin{aligned} d_{t-1} &= \boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p}) \\ \Delta d &= \boldsymbol{n}^T \Delta \boldsymbol{x} \\ d_t &= d_{t-1} + \Delta d \geq 0 \\ \boldsymbol{n}^T \Delta \boldsymbol{x} &\geq -\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p}) \end{aligned}$$

where Δd is the change of the signed distance. The constrained motion can be realized by setting $\boldsymbol{A} = \boldsymbol{n}$ and $\boldsymbol{b} = -\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p})$. The constrained least-squares problem may then be solved to

produce the desired motion. Additional terms may be added to further constrain the tool motion.

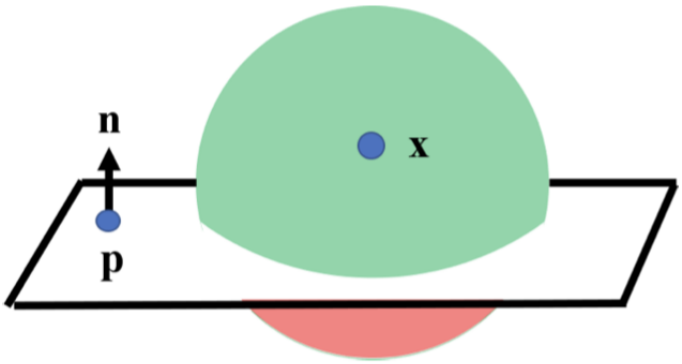


Fig. 1: Illustration of plane constraint. The radius of the sphere is the maximum motion capable of the robot in one control iteration. Green zones denote the allowable region and red zones denote the forbidden region.

3.2 Polygon Mesh

In this work, polygon meshes consisting of triangles are used to represent anatomical surfaces. A locally concave surface (Fig. 2a) produces a convex set of linear constraints. It is safe to include all triangles as plane constraints (Fig. 2b). A locally convex surface (Fig. 2c) produces a non-convex set of linear constraints. Naively adding all triangles as plane constraints will rule out many allowable regions (Fig. 2d). This necessitates an approach to dynamically activate and deactivate the constraints based on the local convexity and concavity of the anatomical surfaces.

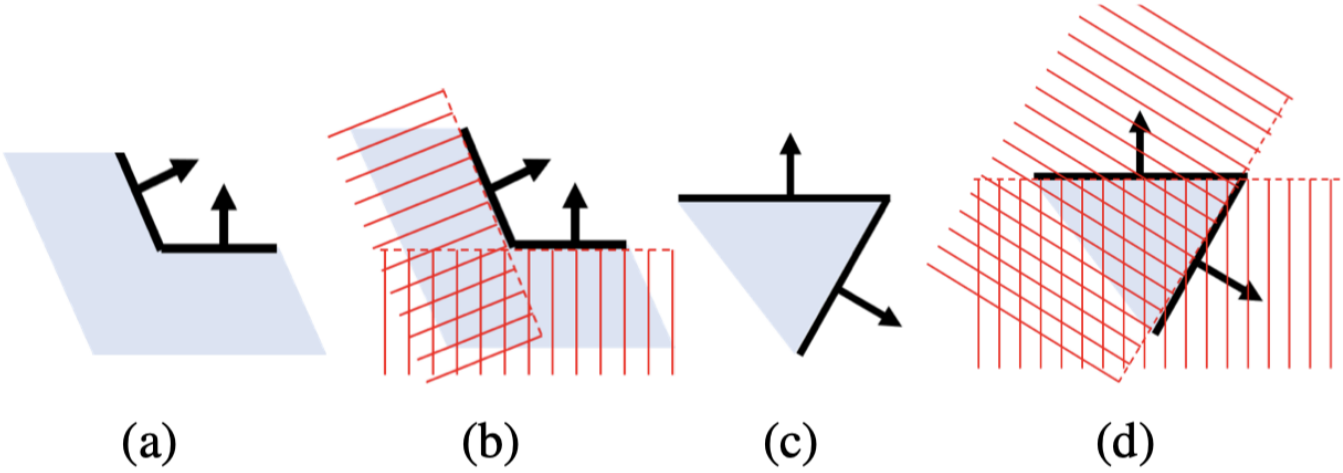


Fig. 2: 2D illustration of (a-b) a locally **concave surface**, and (c-d) a locally **convex surface**. Red crosshatched zones denote forbidden region. Note in (d) many allowable regions are ruled out. Arrows denote the face normals. Light blue zones denote the patient's anatomy.

3.3 Polygon Mesh Constraint Alogrithm

Instead of considering the whole mesh object, a motion sphere is built around the current position with the radius defined by the maximum motion capable of the robot in one control iteration (shown as the sphere in Fig. 1). The triangles intersected by the motion sphere are the only necessary ones to be considered in the current iteration. To enable efficient geometric-search intersection, the anatomical mesh is stored as a Principle Direction Tree (PD-Tree) [14]. The PD-Tree is similar to the KD-Tree, but with nodes split along the maximum distributive direction of the data. This provides an additional boost in search efficiency especially when triangles are not uniformly distributed. During the PD-Tree construction, the adjacency information of the triangles is stored. Using the PD-Tree of the anatomy and the motion sphere of the tool position, the corresponding closest points CP and face normals N of each intersected triangle are returned.

We use the Polygon Mesh Constraint Algorithm developed in [1] to generate proper anatomical constaints for the motion planner. And we use constraint optimization algorithm to command the robot to stop in close proximity to the patient's tissue.

The algorithm is described as follows,

Algorithm 1: Polygon Mesh Constraint

Input: PD-Tree, Current Position x
Result: List of Active Plane Constraints $\mathcal{L}$
Find intersected triangles $\mathcal{T}$, corresponding closest points $\mathcal{CP}$ and face normals $\mathcal{N}$;
for *triangle* $\mathcal{T}_i \in \mathcal{T}$ **do**
  if $\mathcal{CP}_i$ *in-triangle* & $\mathcal{N}_i^T(x - \mathcal{CP}_i) \geq 0$ **then**
    add $\{\mathcal{N}_i, \mathcal{CP}_i\}$ to $\mathcal{L}$;
  else if $\mathcal{CP}_i$ *on-edge* **then**
    Find adjacent triangle(s) $\mathcal{T}_{i,a}$;
    if $\mathcal{CP}_i == \mathcal{CP}_{i,a}$ & *locally convex* **then**
      add $\{x - \mathcal{CP}_i, \mathcal{CP}_i\}$ to $\mathcal{L}$;
    else if $\mathcal{N}_i^T(x - \mathcal{CP}_i) \geq 0$ & *locally concave* **then**
      add $\{\mathcal{N}_i, \mathcal{CP}_i\}$ to $\mathcal{L}$;
end

3.4 *Feature 1: Extending the tool tip point to a sphere

Above we assumed that the tool tip is a geometric point, this holds true for the sharpness of most of the surgical tools, but this isn't inclusive. There exists some surgical tools with blunt tip. Therefore, with regard to generality and compatibilty, it'll be better to use a sphere to model the shape of the tool tip, since with the sphere model, we can have a radius variable to adjust to the sharpness of the tool tip. This can be achieved through modifying formula 2 as follows.

$$d_t = d_{t-1} + \Delta d \geq r$$
$$\boldsymbol{n}^T \Delta \boldsymbol{x} \geq -\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p}) + r$$

We need to change the code implementation to add the sphere model feature.

3.5 *Feature 2: Adding Slack Variable

During the surgery, surgeons sometimes need to violate some virtual fixture constraints of the patient's tissue in order to perform cutting or drilling operation. In order to implement this, we nee to set a slack variable for every active constraint to represent to what extent it can be violated. If some certain virtual fixture constraints are violated in the current planning-and-moving iteration, it should be recorded and automatically added to the next iteration as active constraints, since it might not be recognized as active constraints by the activeconstraint-selection algorithm.

$$\begin{aligned} \boldsymbol{n}^T \Delta \boldsymbol{x} &= -\boldsymbol{n}^T (\boldsymbol{x} - \boldsymbol{p}) + r + s \\ s &\geq 0 \end{aligned}$$

3.6 *Feature 3: Tool Shaft Modeling

Currently, we've only considered the tool tip, either modeling it as a point or a sphere. However, practically, the whole tool shaft should all be considered in the motion planning algorithm. Based on the current point-based algorithm, we propose a sampling based method by which the current algorithm can be extended to the scope of the whole tool shaft.

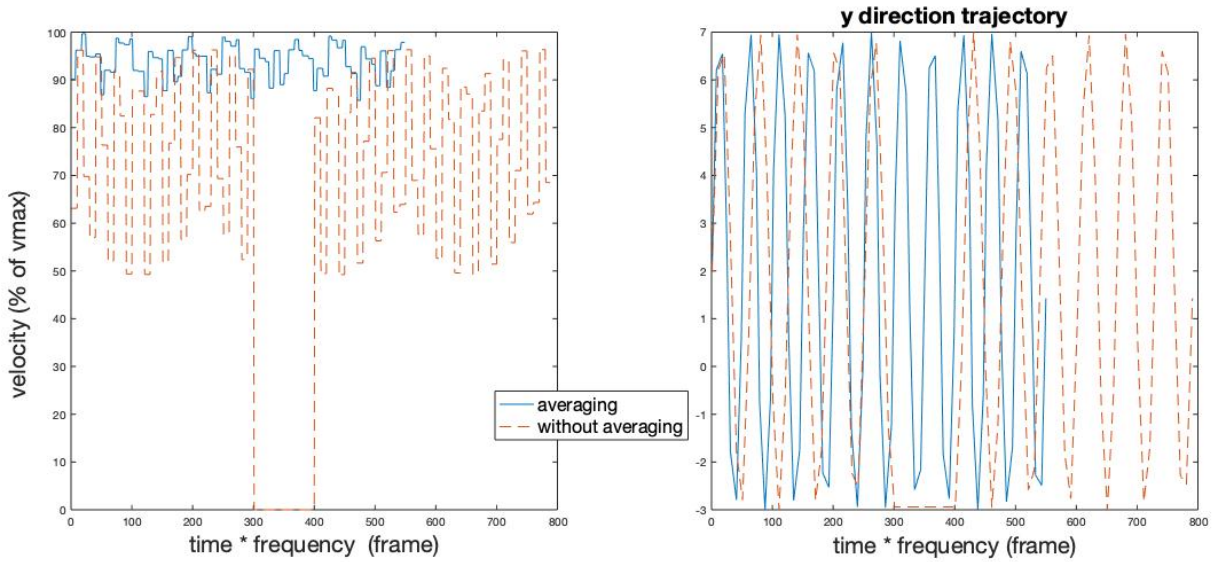
We plan to generate some points evenly distributed in the region of the tool shaft and generate a sphere at each sample points. The radius of each sphere can be adjusted to finely fit the shape of the tool shaft. But for validation, we will only consider the simplest case, model the tool shaft as a cylinder. We will attach 3 rotation variables to the state x, making it 6-Dof. We'll have to adjust the Jacobian matrix accordingly, and modify the construction of the constraint matrix, enabling it to take all the active constraints of all the sample points, with regard to 6 state variables.

The intermediate distance of sample points is deliberately chosen to be 0.88*sphere_radius so that the sampling density is optimally balanced between the sampling efficiency and the gap between two sample spheres.

3.7 *Feature 4: Automatic Tool Retraction

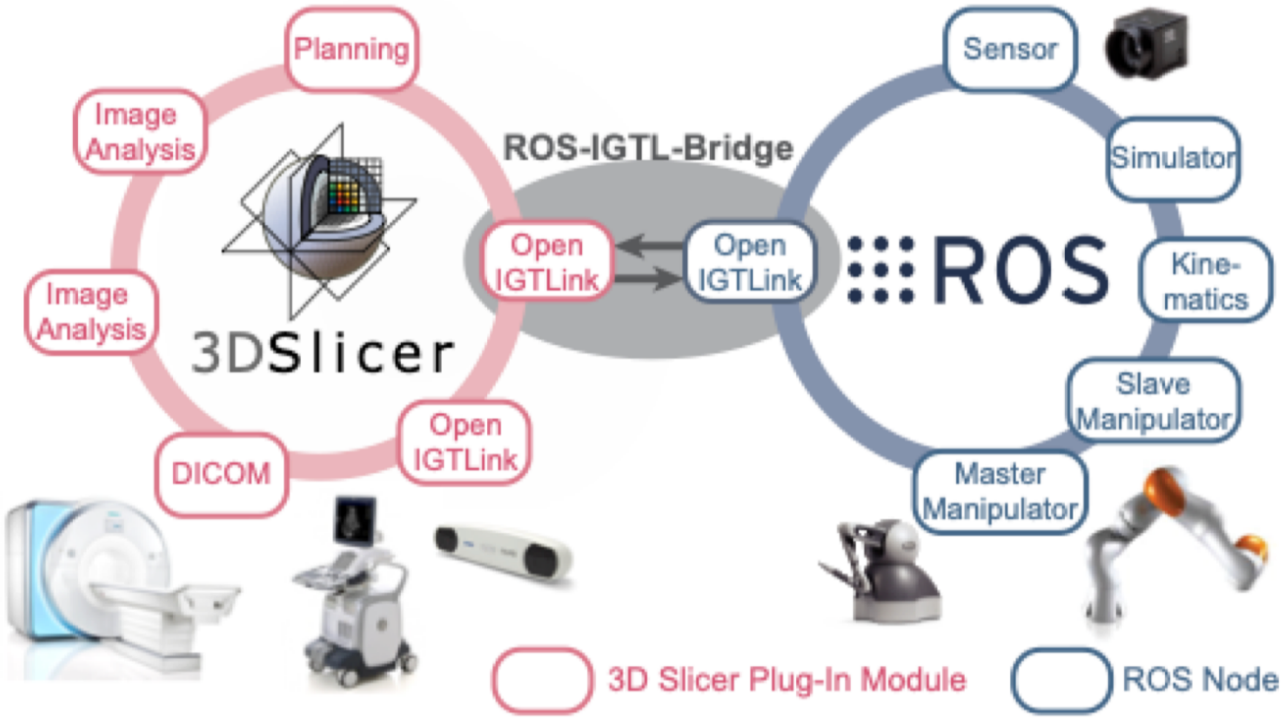
In many surgical navigation scenario, when surgeons have inserted a tool into patient's body, the retraction process can be automated by following the same path and can be accelerated by interpolating and smoothing the 1st order of insertion trajectory. This automatic tool retraction feature can save surgeons some energy and shorten the overall surgery time. This idea is implemented by first recording the insertion trajectory, interpolating intermediate points with various density according to the time stamp of the recorded trajectory points, and reversing the whole sequence, playing from end to the beginning.

Following is the position and velocity trajectory of the retraction process, compared to that of the insertion process.

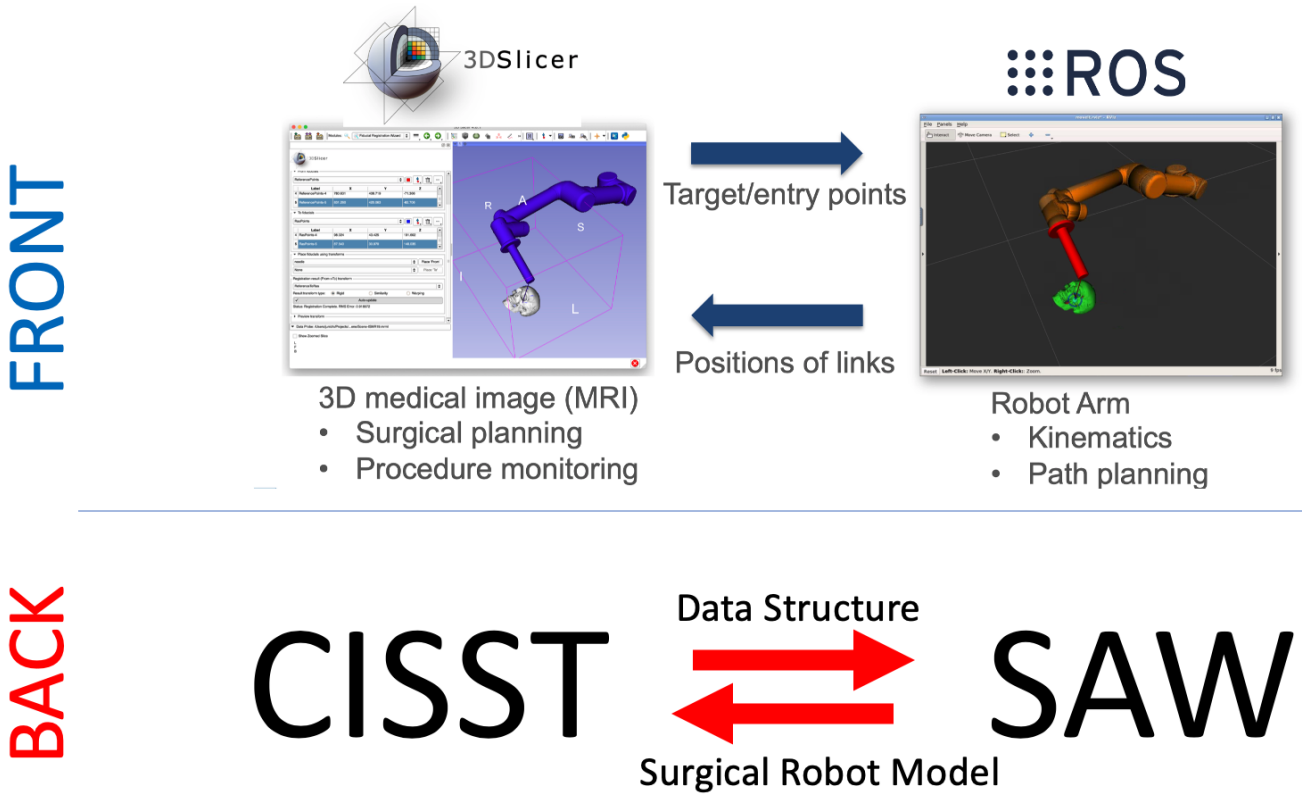


3.8 Simulation System

For front-end visualization， we choose the 3D Slicer software which is dedicated to image-guided robot-assisted interventions. It can commnunicate with ROS system through a protocol called IGTL bridge, which is depicted below.



And Since all the back-end functions are based on the CISST and SAW library, which offer ROS communication interface, the front-end and back-end are compatible. The complete Architecure is shown below.



IV. Management Summary

Below two versions of steps and milestones are kept. One original version, representing the idealistic plan, which demonstrate the ultimate goal of the project and is still expected to accomplish beyond the scope of this course. One revised version, representing the compromised plan due to 2019-nCov situation.

4.1 Original Version

Step 1: Getting started

- Write proposal, give presentation and construct the wiki page.
- Get familiar with the CISST code base (focused on numerical environment)

  Milestone: Complete the task of loading mesh stl binary file and add it to the library. (ASCII file version is already there)

  Date: Mar. 5 ~ Mar. 15 (10 days)

  Status: 100%

Step 2: Integrate Mesh Constraint to Galen Robot

- Learn the usage of Galen Robot.
  Milestone: Test the existing code examples on Galen Robot.
  Date: Mar.16 ~ Apr. 5 (20 days):
  Status: 100%

Step 3: Test the constraint formulation

- Integrate the constraint formulation with simple geometry obstacle
- Integrate the constraint formulation with 3D phantom of patient anatomy.
- Complete the user study (if possible)
  Date: Apr. 6~ May 1 (25 days)
  Milestone: a video to demonstrate the Galen robot can actually perform the above tasks.
  Status: 0%

4.2 Revised Version

The focus has shifted from creating the real-world demo to strengthening the robustness of the mesh constraint algorithm and creating a demo in simulation.

Step 1: Getting started

- Write proposal, give presentation and construct the wiki page.
- Get familiar with the CISST code base (focused on cisstNumerical, and the SAW Constraint Controller part)
  Milestone: Complete the task of loading mesh stl binary file and add it to the library. (ie. feature 1)
  Date: Mar. 5 ~ Mar. 19 (14 days)
  Status: 100% (Step 1 is fully fulfilled, feature 1 is already integrated into the CISST code.)

Step 2: Integrate Mesh Constraint Formulation to Galen Robot Controller Code

- Get familiar with the cisstICP
- Adding slack variable of soft constraints to the optimal controller. (ie. feature 2, this is also the minimum deliverable)
  Milestone: Pass compilation of the integrated Galen Robot cotroller.
  Date: Mar.20 ~ Apr. 5 (14 days):
  Status: 90% (Step 2 is fully fulfilled, feature 2 is implemented but simuilation tests are still going on. Potential bugs could exist.)

Step 3: Test the integrated controller in simulation, using simple robot model

- Test the controller with a simple robot model (eg. UR5 robot arm)
- Test the controller with 3-Dof translation-only motion.
- Test the controller with simple geometry obstacle.
- Test the controller with 3D phantom of patient anatomy. (online data source)
- Improving the mesh-constraint algorithm with respect to the whole surface (use sphere model) of the end-effector, rather than modeling it with a point. (This is also the expected deliverable)
- Test the controller with 6-Dof motion (feature 3).

  Date: Apr. 6 ~ Apr. 20 (14 days)

  Milestone: a video to demonstrate a simple robot can actually perform the above tasks in simulation.

  Status: 20% (Tests are still going on due to the complexity of the simulation environment.)

Step 4: Test the integrated controller in simulation, using Galen Robot model

- Integrating the tested controller into the Galen Robot code base.
- Test all the above tasks with respect to the Galen Robot's forward kinematics and Jacobian matrices.

  Date: Apr. 21 ~ Apr. 30 (10 days)

  Milestone: a video to demonstrate the Galen robot can actually perform the above tasks in simulation.

  Status: 0% (It cannot be fulfilled since the Galen robot's visualization model isn't ready, the. Instead, feature 4 is proposed and implemented.)

V. Deliverables

Likewise, here we kept 2 versions of deliveralbes, idealistically original plan A and relistically revised plan B.

5.1 Original Version

- **Minimum**: Simple geometry code integration
- **Expected**: Patient anatomy code integration(3D phantom)
- **Maximum**: Complete the user study.

5.2 Revised Version

Since the laboratory access is not achievable any more, the goal for the CIS2 project has been shifted to simulation environment accordingly, and improving its robustness, rather than creating a demo in real world.

- **Minimum**: Implement feature 1.
- **Expected**: Implement feature 2/3.
- **Maximum**: Implement feature 4.

All 4 features have been written and passed compilation (code can be found in the appendix). Tests are still going on due to the complexity of the simulation environment. After all the tests are done, a simulation demo will be created and a push request to the CISST code base will be made. (by the end of May)

VI. Dependencies

Item	Contact	Alternatives	Status	Notes
Galen Robot Access	Max	**None**	N/A	Due to CoVID, plan change accordingly
A computer	**My PC**	Dr. Taylor	PlanA succeeded	Hardware
CISST-SAW Code Base Access	**Max**	None	PlanA succeeded	Data Structure & Surgical Robot Model
Preoperative CT scan model	**Pete**	CT scan model from Internet	PlanA succeeded	Testing Data (facial nerve)
CISST ICP	**Max**	None	PlanA succeeded	An Independent Module
Slicer + ITGL + ROS	**Max**	None	PlanA succeeded	Visualization Tool
Galen Robot Model	Can	**use UR5 model**	PlanB succeeded	For demo, Visulization and Debugging

VII. Project Bibliography

1. Zhaoshuo Li et al. Anatomical Mesh-Based Virtual Fixtures for Surgical Robots (unpublished by Mar. 2020)
2. Funda, J., Taylor, R. H., Eldridge, B., Gomory, S., & Gruben, K. G. (1996). Constrained Cartesian Motion Control for Teleoperated Surgical Robots. Robotics, 12(3).
3. Xia, T., Kapoor, A., Kazanzides, P., & Taylor, R. (2011). A constrained optimization approach to virtual fixtures for multi-robot collaborative teleoperation. IEEE International Conference on Intelligent Robots and Systems, 639–644. https://doi.org/10.1109/IROS.2011.6048816
4. Li, M., Ishii, M., & Taylor, R. H. (2007). Spatial Motion Constraints Using Virtual Fixtures Generated by Anatomy. 23(1), 4–19.
5. Kapoor, A. (2008). Motion constrained control of robots for dexterous surgical tasks.
6. S. A. Bowyer, B. L. Davies and F. Rodriguez y Baena, "Active Constraints/Virtual Fixtures: A Survey," in IEEE Transactions on Robotics, vol. 30, no. 1, pp. 138-157, Feb. 2014, doi: 10.1109/TRO.2013.2283410.
7. Masotidectomy Video https://www.youtube.com/watch?v=jnonLwxW2Cg
8. 3D Slicer Simulation Environment Tutorial https://rosmed.github.io/ismr2019/prerequisite
9. CISST-SAW Code Base https://github.com/jhu-cisst/cisst/wiki
10. cisstICP Code Base https://git.lcsr.jhu.edu/zli122/cisstICP
11. VF guided skull cutting using ultrasonic cutter and dVRK https://github.com/mli0603/PolygonMeshVirtualFixture
12. Galen Model Base https://bitbucket.org/GalenRobotics/researchrepo/src/master/source/robot/
13. STL reader https://github.com/sreiter/stl_reader
14. STL parser http://www.dillonbhuff.com/?p=5

VIII. Research Log

- Weekly Meeting Log

IX. Appendix

- My Code Base