High Dexterity Intraocular Manipulation

CIS II Project Final Report

Kaiyu Shi & Yishun Zhou

May 6th, 2021

Table of Contents

1. Abstract

A system to cooperatively control two robots to achieve high dexterity manipulation for vitreoretinal surgery was developed and tested in simulation. The system was able to receive positional and orientational input from a Phantom Omni device. It was also capable of following a pre-planned trajectory. The target goals are constrained such that the position and orientation reached by the manipulator's end effector do not exceed the space of the vitreoretinal space. Furthermore, a force model was formulated to estimate the force on the end effector. FBG (Fiber Bragg Gratings) force sensors in future iterations of the system could halt the system if force on the end effector or sclerotomy exceeds critical thresholds.

2. Clinical Motivation

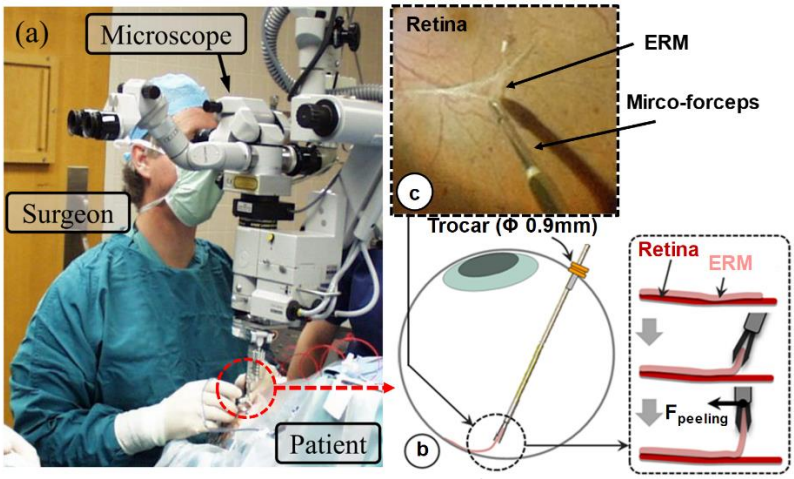


Figure 1: ERM peeling

Vitreoretinal surgery requires advanced surgical skills at or over the limit of surgeons' physiological capabilities. The surgery is performed in a confined intraocular space with restricted free motion of surgical tools. The forces exerted between the ophthalmic tools and eye tissue are often well below human sensory thresholds [1]. For instance, epiretinal membrane (ERM) peeling surgery shown in Figure 1, where a micron-scale membrane on the retinal surface is removed, requires the forces exerted by the surgeon to be less than 7.5 millinewtons. And a force above the 7.5 mN could cause irreversible damage to the retina [2].

This project aims to move towards solving the clinical challenges mentioned above by providing the surgeons a cooperatively controlled robotic system with 2 DOF snake-like manipulator and a 5 DOF Steady Hand Eye Robot that has the capabilities of 1) tremor-free tool manipulation, 2) increased dexterity to ensure safe access to target from suitable directions, and 3) force sensing at the tool tip and sclerotomy.

2. Prior Work

This project will build upon prior work conducted by students and faculty at JHU. Key components include the Steady Hand Eye Robot (SHER), the Integrated Robotic Intraocular Snake (I^2RIS), and a multi-function force sensing and variable admittance control algorithm.

Steady Hand Eye Robot (SHER)

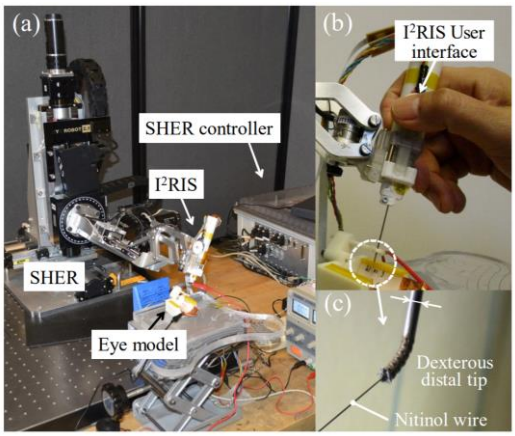


Figure 2: The integrated hardware with both SHER and I^2RIS and experiment setup.

The Steady Hand Eye Robot [3] has five actuated Degrees of Freedom (DoF) and one passive DoF (tool rotation about tool axis), which are comprised of three translational DoFs (X,Y,Z) with linear stages and two rotational DOFs (roll and pitch), as shown in Figure 3 [4]. The robot is carefully designed with a mechanical RCM pitch mechanism so that surgeons can insert surgical tool into the intraocular space through sclerotomies (RCMs) and perform procedures including pitch and yaw motions of the tool without much movement at the sclera contact point which could cause tissue damage.

Integrated Robotic Intraocular Snake (I^2RIS)

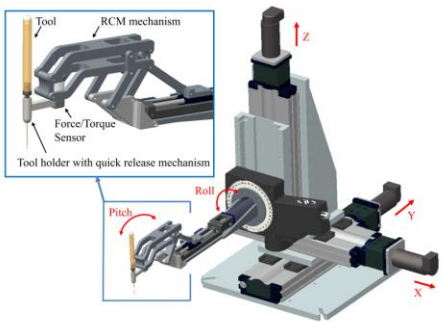


Figure 3: CAD model of SHER and 5 DOFs

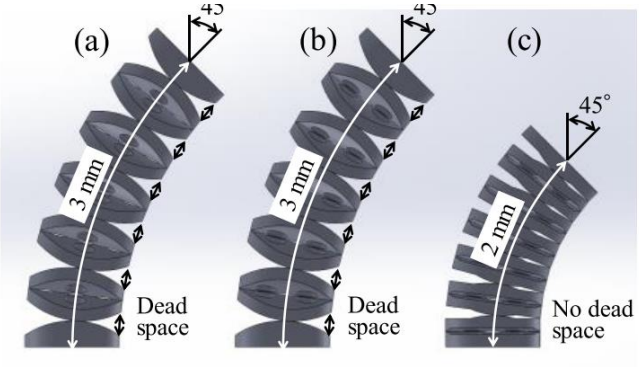


Figure 4: IRIS and I^2RIS: (a) conventional element, (b) conventional proposed element, (c) compact proposed element.

Several design iterations of intraocular snake robot have been developed and tested [1]. The most compact I^{2}RIS design, that we will base our project on is shown in Figure 4 (c). It has a diameter of 0.9 mm and a length of 2 mm with a ±45° bending motion range. And it will provide two DOFs (pitch and yaw) actuated by four wires on a drive pulley perpendicular to the actuation direction. The relationship between the control input, the rotation of drive pulley, and the control output, the bending angle of the snake-like distal end, was determined by geometric model and confirmed by experiments.

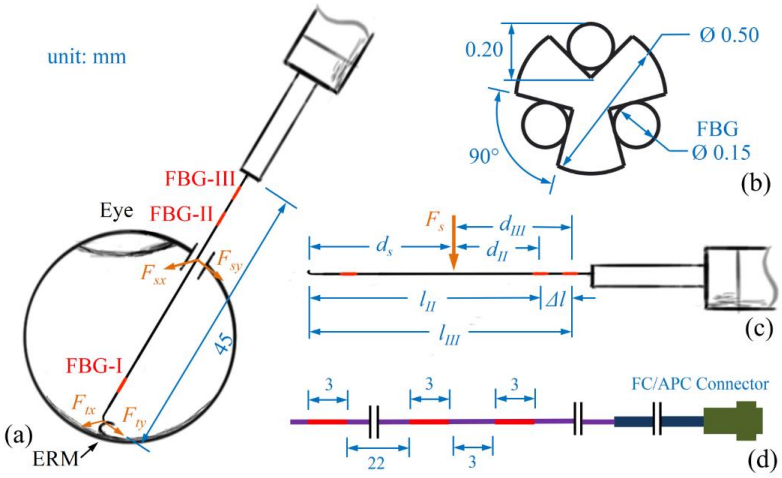


Figure 5: Tool shaft with FBG sensors

A multi-functional force sensing design was built using two sets of FBG segments, each with 3 FBG sensors [5]. The linear mapping from FBG sensor to forces applied was found from calibration experiments. Based on the assumption that the surgical tool is a rigid straight instrument when transversal forces are applied on tool tip and the shaft as Figure 4 shows, a force distribution model was established so that the forces at instrument tip and sclera and the sclera contact position can be calculated from FBG sensor readings.

A variable admittance control algorithm of SHER was designed to allow hand-over-hand control by the surgeon and to provide sclera force feedback to adapt to possible movement of the RCM point during surgeries [5].

3. Problem and Goals

It has been established that the I^{2}RIS and SHER systems are effective at positioning the end tool for robot-assisted Vitreoretinal surgery. However, the independent systems of control proved to be cumbersome in terms of actual use [1]. Additionally, the clinical motivation demonstrates that is both desirable and possibly necessary to have force constraints on the system, to reduce potential trauma on the patient. To these ends the follow goals were formulated to address the present issues:

- Cooperatively control SHER robot and I^{2}RIS snake robot in simulation
- Tool tip follows a desired trajectory while maintaining force constraints (as deduced from FBG readings) on both the sclerotomy and tool tip
- Surgeon controls the 7 DOF combined system with a 5 DOF Phantom Omni device

Successful in achieving these goals would mean a surgeon would be able to conduct micro-surgery such as ERM peeling efficiently, without hand tremors, and with the confidence that the force being applied is within acceptable thresholds.

4. Solution

Assumptions

To approach the problems and goals in this project, we set up several assumptions. We assumed that the robot is rigid and the dimensions from CAD are perfect to eliminate noise from vibration and the need for calibration. We assumed that the intraocular space is a sphere that is perfectly registered to the robot, and that the sclerotomy incision point for which the surgical tool goes through is also perfectly registered to the robot. This allowed us to develop the system by first resolving systematic issues. Future work can remove these assumptions to further improve the system for real world application.

Simulation

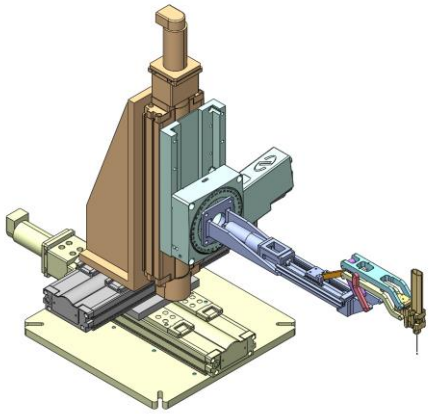


Figure 6: CAD Model

The combined CAD model was generated from files generated from prior work. The model is used as a reference for the kinematics and simulation representation. See below for Forward Kinematics derived from the CAD model.

Additionally, a 3D model of an eyeball was created to represent an 3D phantom for the manipulator to operate in. It uses a spherical cavity to approximate the vitreoretinal space.

The simulation is created as a ROS package, *gazebo_sim,* with the robot system described for Gazebo world using the URDF description format. The system's 7 degrees of freedom is enabled by the 7 joints q1-q7, which are each controlled by a velocity_position_controller via the gazebo_ros_plugin. The target values for the joints are exposed as ROS topics. Since URDF does not allow for closed-loop linkage systems, so the four bar linkage uses multiple mirrored joints (q5 – q5e) which represent the single joint q5. For the snake manipulator, the two degrees of freedoms are abstracted as the joints q6 and q7, which represent 22 joints (q6a – q6k, q7a – q7k) that all mimic the same command rotation. This enables kinematically accurate simulation of the entire system.

The simulation is launched with the ROS launch file gazebo_model.launch.

Additionally, for faster debugging purposes, a Rviz visualization was also created, and can be launched with the launch file rviz_model.launch.

Interface with Phantom Omni

When the user moves the phantom omni device [7], the input pose is published in the "/rostf" topic, from which we can extract the transformation of the Omni from its base to tip. We can then scale this input pose from the Omni space, which has an approximate diameter of 0.13 m, to intraocular space, which has an approximate diameter of 0.015 m. The scaling ensures a higher resolution for user input.

Frames of Reference

For ease of reference, multiple frames of reference were used to make it easy to perform relative additions and subtractions, as well as to constrain certain values such as insertion distance.

The eye frame uses the sclerotomy as the origin and uses the following the describe the configuration of the robot: *q_eye* = [*roll pitch dist s_yaw s_pitch*]. The *roll* and *pitch* are that of the SHER robot, the *dist* is the insertion distance of the end of the rod past the sclerotomy point, and *s_yaw* & *s_pitch* represent the snake manipulation.

The robot frame is defined as the global frame of the world, translated such that the origin lies on the sclerotomy, but moves with the 3 axis linear movements of the robot. This frame uses: *q_robot* = [*x y z roll pitch s_yaw s_pitch*]

Lastly, the global frame is used to calculate the joint values for the linear axis motors.

To transform between the different frames of reference, the functions *Robot_Eye_q()* and *Eye_Robot_q()* are used. The offset between the robot frame and global frame, which is the distance from initial RCM point to the sclerotomy, is calculated at runtime once the sclerotomy point is registered.

Forward Kinematics

Forward Kinematics of SHER

SHER has a total of 5 joints, including x, y and z translations and rotations around y-axis and x-axis. These joints are called q_1 to q_5 respectively. We will use the notation F= [R,p] for transformation and Rot(axis,angle) for rotation. The following forward kinematics model agrees with the URDF description of the robot.

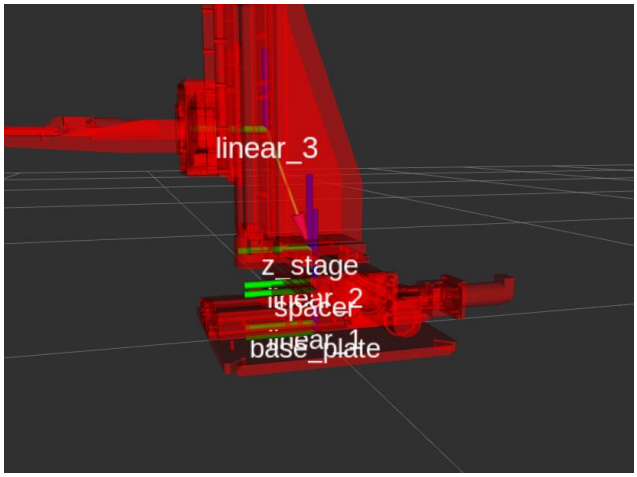


Figure 7 Frames including X and Y Stages

From ”base_plate” to ”linear_1”, F_1= [I,(0,0,0.0127)]

From ”linear_1” to ”spacer”, F_2= [I,(0,q_1,0.04725)]

From ”spacer” to ”linear_2”, F_3= [I,(0,0,0.0127)]

From ”linear2” to ”z_stage”, F_4= [I,(q_2,0.0075,0.04725)]

From ”z_stage” to ”linear_3”, F_5= [I,(0,0.058,0.15858)]

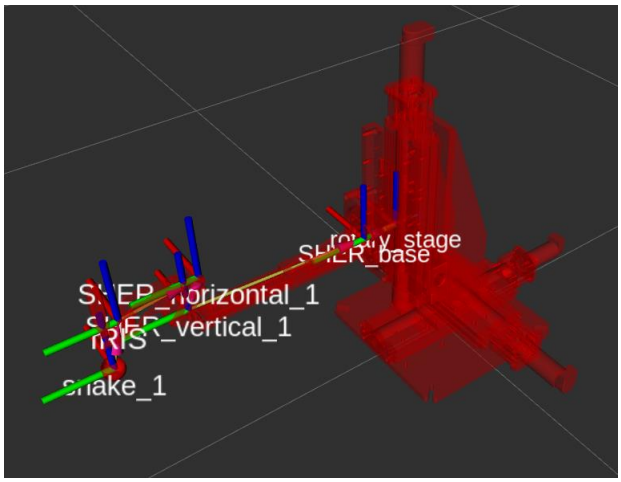


Figure 8: Frames including Z Stages, Roll and Pitch

From ”linear_3” to ”rotary_stage”, F_6= [I,(00.04725,q_3)]

From ”rotary_stage” to ”SHER_base”,F_7= [Rot((0,1,0),q_4),(0,0.063,0)]

From ”SHER_base” to ”SHER_horizontal1”, F_8= [Rot((1,0,0),q_5),(0,0.304,0.015)]

From ”SHER_horizontal1” to ”SHER_vertical1”, F_9= [Rot((1,0,0),−q_5),(0,−0.023,0.048)]

From ”SHER_vertical1” to ”IRIS”, F_{10}= [Rot((1,0,0),q_5),(0,0.120,−0.015)]

From ”IRIS” to ”Snake_1”, F_{11}= [I,(0,0.02177,−0.07628)]

From ”Snake_1” to ”1_2”(first virtual snake joint), F_{12}= [I,(0,0.02177,−0.07628)]

The forward kinematics of SHER is then the multiplication of F_1 to F_{12}.

Forward Kinematics of I²RIS

I²RIS has two input rotational “joints” named q_6 and q_7. These joint angles control the amount and direction of rotation between each two links of the snake. Note that the direction of rotation alternates from link to link and is perpendicular to the last one. q_6 represents the rotation around the y-axis, while q_7 represents the rotation around the x-axis.

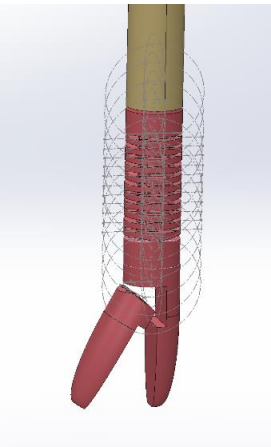


Figure 9: Snake End of I²RIS

There is a spherical face between each two links of the snake. We can construct two virtual circles as shown in Figure 5, which fits the spherical surfaces, to represent rotation between links. We denote the rotation around y-axis R_6=Rot((0,1,0), q_6) and the rotation around x-axis R_7=Rot((1,0,0), q_7).

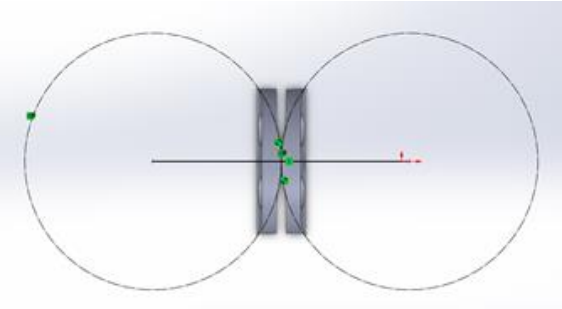


Figure 10: Joint Mechanism of Snake Distal End

The transformation between any two links could be represented as two transformation matrices with the same rotation part (either R_6 or R_7) such as F_{7a}= [R_7, (0,0,0.00145)] and F_{7b}= [R_7,(0,0,−0.0016)].The forward kinematics of the snake would include the multiplication of 12 pairs of such transformations with the first one being [R_6, (0,0,0)T], which is then post multiplied by [I,(0,0,−0.00195)].

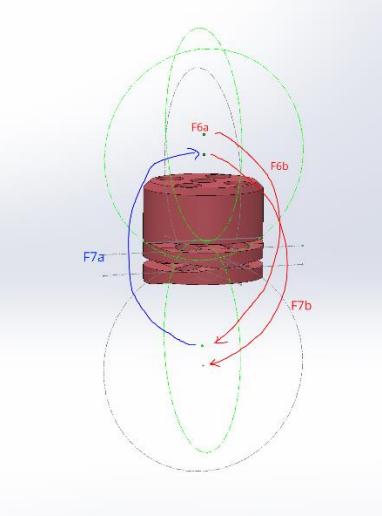


Figure 11: Frame Transformation in the Snake Robot

Figure 11 shows the relative positions of rotational centers between virtual joints of different segments.

Forward Kinematics from Sclera to Tool Tip when Tool is inserted

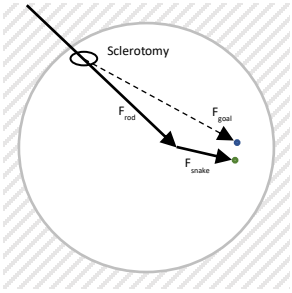


Figure 12: Transformation from the Sclera to Tool Tip with Marked Constraints

We can find the transformation from the sclera to the start of the snake called F_{rod} shown in the equations below by finding the distance of insertion denoted by dist and the roll and pitch angles q_4 and q_5. F_{snake} can be found from the previous section. And the transformation from the sclera to the tool tip is just $F_{rod}*F_{snake}$. Note that this transformation in the intraocular space is equivalent to a transformation from the base of the robot to the tip.

$$F_1 = [Rot((0\ 1\ 0), q_4),(0\ 0\ 0)]$$

$$F_2 = [Rot((1\ 0\ 0], q_5),(0\ 0\ 0)]$$

$$F3 = [I, (0\ 0\ -dist)]$$

$$F_{rod} = F_1 * F_2 * F_3$$

Constraints

When we get a pose input from the user to move the tool tip to a certain position and orientation in the eyeball, there are some constraints that we use to modify the pose input to ensure safety of the patient:

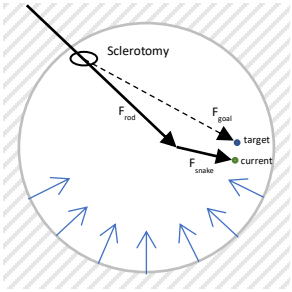


Figure 13: Target Goal Constraints

1. The input is constrained to a target goal within the eye. This is done with *getConstrainedGoal()*.
2. The target goal moves towards the Omni's position proportionally ("P" loop), capped to *max_speed.* This is done in *getNewGoalOmni().*

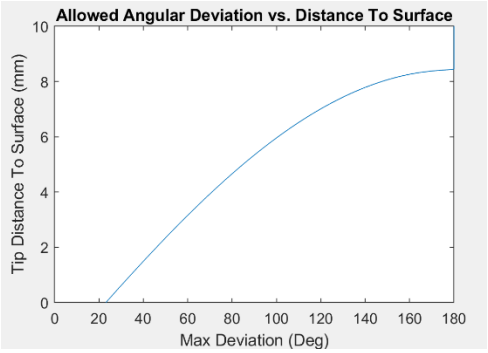


Figure 14: Allowed Angular Deviation From Surface Normal

1. The orientation is constrained towards the surface normal as a function of the distance from the surface. This is also done with *getConstrainedGoal()*.

When solving the optimization problem, the following constraints are applied:

2. The joints are limited to predefined ranges. This is done with *q_limits().*
3. The change in joints is limited to the maximum angular/linear velocities. This is done with *dq_limits()*. Notes: the limit values are currently placeholders.
4. The I^2RIS robot is constrained to pass through the sclerotomy. This is implicit with the conversion between eye and robot frames of reference.

Optimized Inverse Kinematics

To solve for the target joint values at the next step, an optimization problem [6] is solved iteratively within the eye frame of reference (relative to the sclerotomy point):

$$while\ error_{pos} > 0.05\ mm\ and\ error_{rot} > \sim 1.5°:$$

$$Jacobians = [Jacobian_{pos}(q_{eye}); Jacobian_{rot}(q_{eye})]$$

$$X_{curr} = FwdKin_{eyeorigin\_tip}(q_{eye})$$

$$Delta = X_{goal} - X_{curr}$$

$$error_{pos} = norm(Delta(1{:}3))$$

$$error_{rot} = norm(Delta(4{:}6))$$

$$argmin_{\Delta q} \| Jacobians * \Delta q - Delta) \|$$

$$q = q + \Delta q$$

$X_{curr} = [x\ y\ z\ e_1\ e_2\ e_3]$ is the position and orientation of the current position, while X_{goal} is the goal position and the orientation.

Because the DoF of the robot within the vitreoretinal space is not sufficient to provide the full 3 DoF for all possible rotations, we do not represent the orientation with Euler angles, as the Jacobian generated from that would not be suitable for maneuvering the end effector to a solution that is minimizing the joint movement. Instead, the orientation is represented as a unit vector pointing in the direction of the end effector, which allows the end effector to be orientated without consideration of the "roll" of the end effector. The rotated unit vector is calculated by rotating a unit vector (i.e. [0 0 -1]) by the rotation matrix of the forward kinematics transformation. The input orientations and generation of the rotation Jacobian also use this formulation.

The Jacobians are pre-generated symbolically in MATLAB in *generateEyeJacobian.m* and loaded in at runtime to reduce overhead in solving the inverse kinematics problem.

$$Jacobians = \frac{\delta \begin{bmatrix} x(q_{eye}) \\ y(q_{eye}) \\ z(q_{eye}) \\ \vec{e}(q_{eye}) \end{bmatrix}}{\delta q_{eye}} = 6 \text{ x } 5 \text{ matrix}$$

The *argmin* problem is solved to find a least-squares solution for Δq by using the MATLAB *linsolve()* function.

Force Model

Overview of the force sensing tool

The force sensing tool contains three FBG sensors, one near the inner membrane of the eyeball and the other two outside the eyeball. The placement of the sensors are fixed as shown in Figure ??. And the FBG sensors are approximated as force/torque sensors limited in two directions (x and y). We denote the measurement from the sensor $\Delta S_j = [\Delta s_{j1}, \Delta s_{j2}, \Delta s_{j3}]^T$, where j=I, II, III.

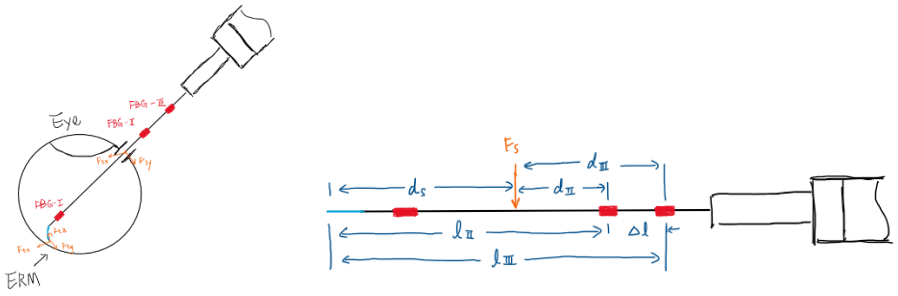


Figure 15: Force Sensor Diagram

Force on Tip

The force induced on FBG I is proportional to the sensor readings:

$$\Delta S_I = K_I F_I, F_I = \left[F_{Ix}, F_{Iy}\right]^T$$

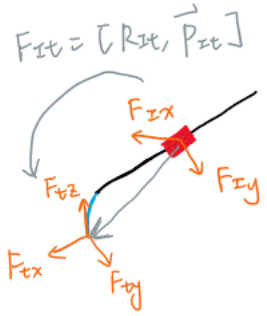


Figure 16: Force at Tip

From the kinematics model, we can obtain the transformation matrix from FBG I to snake tip $F_{It}=[R_{It},p_{It}]$. From FEA analysis, the force at the tip F_t should equal to F_I. Using the information above, we can obtain the following equations:

$$\vec{F_t^I} \cdot \vec{x_I} = (R_{It} \cdot \vec{F_t}) \cdot \vec{x_I} = F_{Ix}$$

$$\vec{F_t^I} \cdot \vec{y_I} = (R_{It} \cdot \vec{F_t}) \cdot \vec{y_I} = F_{Iy}$$

Force at the Sclerotomy

The torque induced at the sensor FBG II and III should be proportional to the sensor readings:

$$\Delta S_j = K_j \tau_j, \tau_j = \left[\tau_{jx}, \tau_{jy}\right]^T, j = II, III$$

Both forces at the tip and near the sclerotomy contribute to the torque at the sensors II and III. Sowe have the following equations:

$$\vec{\tau_j} = \overrightarrow{\tau_t^j} + \overrightarrow{\tau_s^j}, j = II, III$$

Similarly to the precious section, we can find the transformation matrix from the FBG sensors to tip, F_{IIt}= [R_{IIt},p_{IIt}] and F_{IIIt}= [R_{IIIt},p_{IIIt}]. We could then find:

$$\overrightarrow{\tau_j} = \overrightarrow{p_{jt}} \times (R_{jt} * \overrightarrow{F_t})$$

Then we can find the forces at the sclerotomy from the equations below:

$$F_{sy} = \frac{\tau_{s,1}^{II} - \tau_{s,1}^{III}}{\Delta l}$$

$$F_{sx} = \frac{\tau_{s,2}^{II} - \tau_{s,2}^{III}}{\Delta l}$$

We can also find the distance from the sclerotomy to the sensors:

$$d_j = ||\tau_s^j||/||F_s||$$

Code

The code dependency graph is listed in Appendix A. All components of the code are commented with the input/outputs as well as explanation if needed. Some of the key components of the codebase are as follows:

InvKinOmniControl(): main script for the control system

OmniSub(): subscribes to /tf topic and returns the scaled input goal

InvKinSolver(): solves the inverse kinematics problem for each time step

getConstrainedGoal(): constrains the input goal to “safe” position + orientation

5. Results

Evaluation of Kinematics Model

The validation of the forward kinematics model is done by comparing the calculated end-effector position of the robot given joint angles to that in simulation. Similarly, the validation of the inverse kinematics model is done by comparing the calculated joint angles and translations given end-effector position to that of the robot joints in simulation.

Evaluation of Inverse Kinematics Control

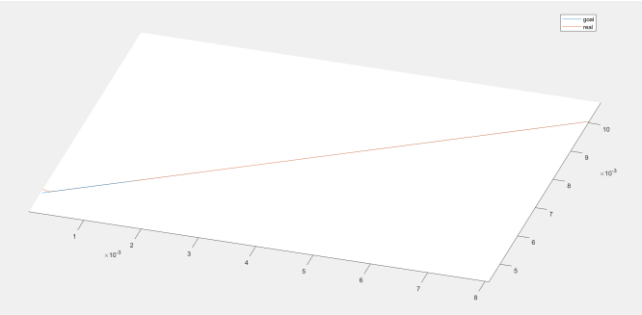


Figure 17: Straight line path traversal

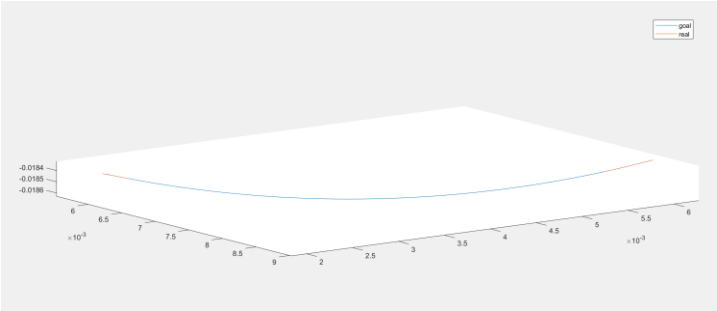


Figure 18: Curved line path traversal

Paths are generated with generate_trajectory() in the form of an array of [x y z $\vec{e}$] values, representing the points along the path. The points are loaded into a queue data structure. During each time step of the control loop, the next goal in the queue is popped and set as the new input goal.

The system was evaluated by following two pre-planned paths and recording the position of the end effector, by subscribing to the /link_states topic of gazebo, and comparing with the recorded target path. The two paths consist of a straight linear traversal (fig. 17) and a traversal along the bottom surface of the eye (fig. 18). It is evident that the end effector position was within the error tolerances set for the optimization loop. However this result is reflective of the "perfect" nature of the simulation, and can only serve to prove that the algorithm does not have systematic error. Additional plots of the traversals are in Appendix B.

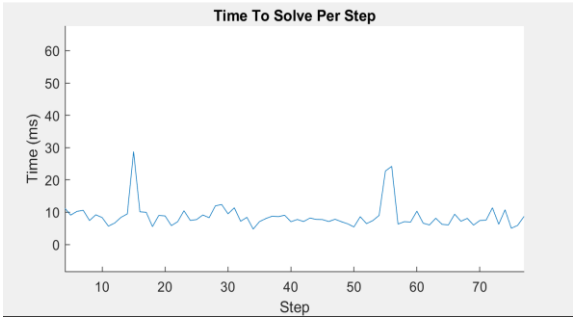


Figure 19: Time to Solve Inverse Kinematics

The time to solve for the inverse kinematics was plotted for each time step for a sample path. Since the times are below 30 milliseconds, it means the system in generally able to operate in real time. However, it is not guaranteed that this is always the case.

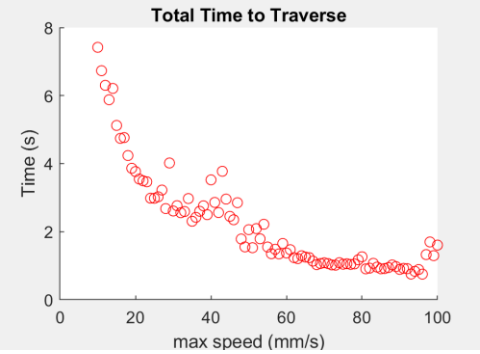


Figure 20: Total Time to Traverse vs. Max Allowed Speed

The time to traverse an identical linear path is plotted against the maximum allowed speed for the end effector. It is evident that there is a non-linear relationship between the maximum speed and the actual time it takes to traverse the path. At roughly 20 mm/s it seems there is a substantial increase rate of growth in traversal time as the max speed of the end effector is reduced.

6. Significance

This work calculated the forward kinematics model of the snake robot I^2RIS and optimization based inverse kinematics of the combined robot system of both SHER and I^2RIS, which is enables concurrent control of the combined robot. The implementation of the cooperative control along with Phantom Omni input enables the surgeon to position the end effector in a naturally intuitive way while maintaining safety for the patient. The MATLAB code we have written in extensible for future improvements.

7. Future Work

Physical Implementation

Since this project was designed to work over the ROS framework, it is possible to switch the joint commands sent out to the simulation over to the physical robot, provided the motor controllers for the robot is also interfacing with the ROS framework. Additionally, some other work would be done to ensure the control system works on a physical robot, such as calibrating the values for the I^2RIS joints.

Force Sensing

Unfortunately, we were not able to validate the force model for this project due to time constraints. In the future this could be done and tested with physical FBG sensors on the I^2RIS.

Trajectory Constraints

Vitreoretinal surgery and epiretinal membrane peeling involve complex trajectories of the forceps. In the future, we could explore desired peeling trajectories and ideal constraints that could reduce tissue damage and lower difficulties of these surgery procedures.

Further Evaluations and Testing

Due to software issues between ROS and MATLAB, we were not able to record the error between goal pose and actual pose of the robot using Omni as input in real-time. This evaluation test can be conducted in the future by using a separate ROS node to record the information. More extensive testing should be conducted to test the control algorithms with more edge cases.

Haptic Feedback

Haptic feedback to the surgeon through the Phantom Omni or a similar device could be useful as a way of conveying information in an intuitive manner.

8. Acknowledgements

This project was funded with JHU internal funding. Many thanks to Professor Iordachita and Dr. Gang Li for mentoring and advising us. Also, thanks to Professor Russell Taylor for providing valuable project feedback.

9. Management summary

Credits

Kaiyu mainly worked on creating the robot simulation, implementing the optimization based inverse kinematics and programming the control of robot. Yishun mainly worked on forward kinematics and implementing control of robot with Phantom Omni.

Dependencies

Dependency	Status	Contingency	Followup	Funding	Deadline	Situation	
SHER	Exists	Simulation	-	JHU Internal Funding	-		✓
I²RIS	No FBG force sensors	Only implement position control/FBG in simulation	Discuss with Prof. Iordachita	JHU Internal Funding	3/29	No FBG sensors	✗
Computer running Linux for simulation	Exists	-	-	Personal computer	-		✓
Phantom Omni	In lab	Joy-stick input/keyboard input	-	JHU Internal Funding	-	Acquired w/ ROS packages & linux laptop	✓

Most of our dependencies required for simulation are met at the beginning in February. We also acquired the Phantom Omni with its ROS packages installed on a linux laptop in April for control of the robot with Omni. We didn't acquire the FBG sensors on the physical robot which are necessary for implementing control of the physical system (maximum deliverable). But time constraints are the main reason why we did not complete the maximum deliverable. In the future, this dependency needs to be acquired to continue this project.

Deliverables

Most of minimum and expected deliverables were completed except the force sensing functionality due to time limitations. The check mark with yellow margin indicates fully completed, check mark with no margin indicates completed with missing functionality and no check mark indicates not completed. Solving for the inverse kinematics model took us longer than expected, which pushed our deadline for other activities to two weeks later. And the robot control part is completed without the force sensor feedback. The maximum deliverables will be the next step.

	Milestones	Planned Deadline	Status	Deliverables	Planned Deadline	Status
Minimum	Forward kinematics model	3/8	✓	A report that includes calculation of forward kinematics	3/8	✓
	Inverse kinematics model	~~3/8~~ 4/10	✓	A report that includes of inverse kinematics, and jacobian of the combined system	~~3/8~~ 4/10	✓
	Force distribution model	3/15	✓	A report on the force distribution analysis	3/15	✓
	Control algorithm pseudocode	~~3/29~~ 4/10	✓	A schematic of the control algorithm design	~~3/29~~ 4/10	✓
Expected	Setup robot model in Gazebo	3/4	✓	A functioning gazebo simulation in which the end-effector of the simulated eye robot follow several optimized trajectories	~~4/5~~ 4/10	✓
	Control algorithm testing and validation in simulation	~~4/5~~ 4/20	✓	A report that summarizes the control algorithm, and an evaluation of the simulated system	5/5	✓
Maximum	Control algorithm testing and validation on real robot	~~4/19~~		Implemented control system on real hardware	~~5/5~~	
				Documentation of implementation	~~5/5~~	

Lessons Learned

Throughout the project, we have been working with robot kinematics, object-oriented programming in MATLAB, robot simulation and communication using ROS, gazebo and rviz. Although our project is implemented in simulation, we still ran into unexpected problems such as inaccurate physics properties in Gazebo and limitations with communication. An important lesson we have learned is that we should leave time for solving such unexpected problems and not assume functionality described will “just work”.

References

[1] Jinno, Makoto, and Iulian Iordachita. “Improved Integrated Robotic Intraocular Snake*.” *2020 International Symposium on Medical Robotics (ISMR)*, 2020, doi:10.1109/ismr48331.2020.9312927.

[2] P. Gupta, P. Jensen, and E. de Juan, “Surgical forces and tactile perception during retinal microsurgery,” in *International Conference on Medical Image Computing and Computer Assisted Intervention*, vol. 1679, 1999, pp. 1218–1225

[3] Uneri, Ali, et al. “New Steady-Hand Eye Robot with Micro-Force Sensing for Vitreoretinal Surgery.” 2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics, 2010, doi:10.1109/biorob.2010.5625991.

[4] Makoto Jinno, Gang Li, Niravkumar Patel, Iulian Iordachita, “An Integrated High-dexterity Cooperative Robotic Assistant for Intraocular Micromanipulation”, 2021., Kokushikan University

[5] He, Xingchi. Force Sensing Augmented Robotic Assistance for Retinal Microsurgery. 2015. Johns Hopkins U, PhD dissertation.

[6] T. Xia, A. Kapoor, P. Kazanzides and R. Taylor, "A constrained optimization approach to virtual fixtures for multi-robot collaborative teleoperation," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 639-644, doi: 10.1109/IROS.2011.6095056.

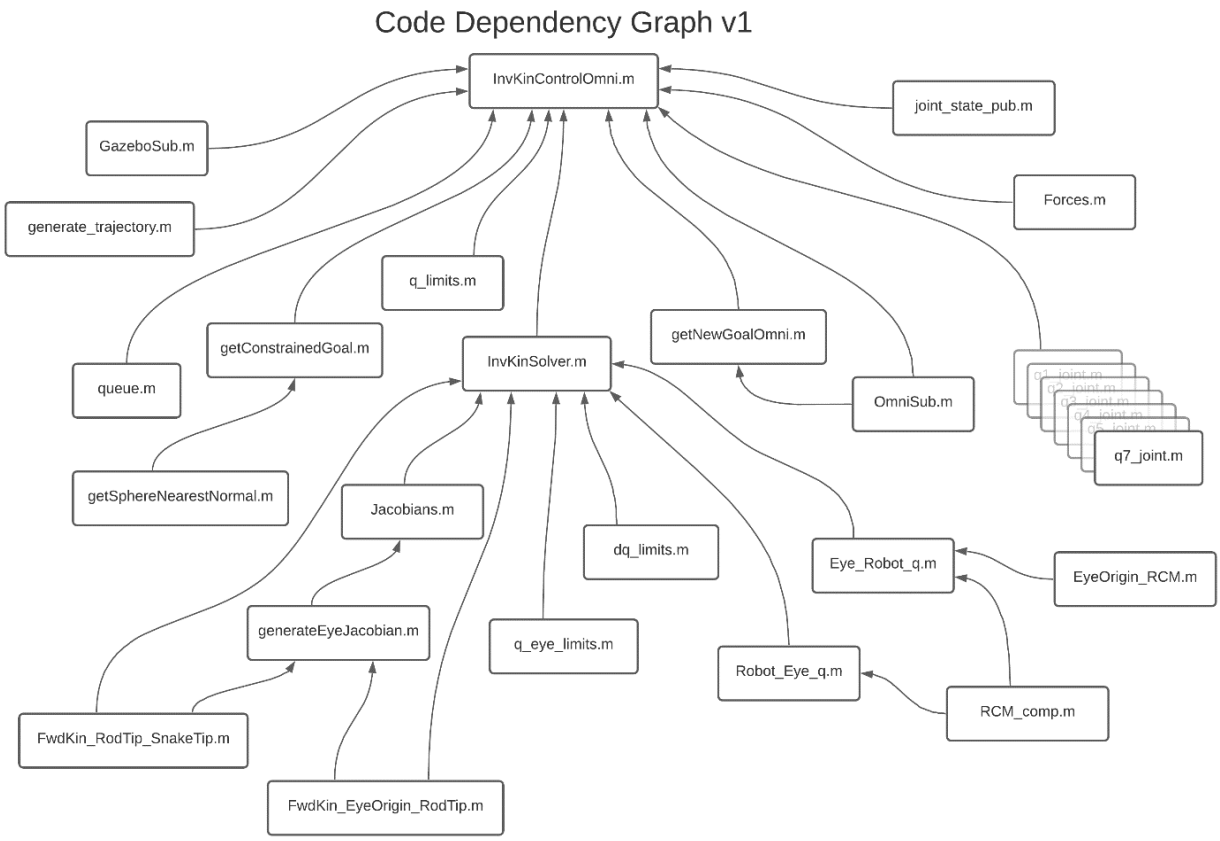
[7] 3D Systems. “Touch.” *3D Systems*, 4 June 2020, www.3dsystems.com/haptics-devices/touch.

[8] Üneri, Ali, Marcin A. Balicki, James Handa, Peter Gehlbach, Russell H. Taylor, and Iulian Iordachita. "New steady-hand eye robot with micro-force sensing for vitreoretinal surgery." In 2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics, pp. 814-819. IEEE, 2010.

[9] M. Li, M. Ishii and R. H. Taylor, "Spatial Motion Constraints Using Virtual Fixtures Generated by Anatomy," in IEEE Transactions on Robotics, vol. 23, no. 1, pp. 4-19, Feb. 2007, doi: 10.1109/TRO.2006.886838.

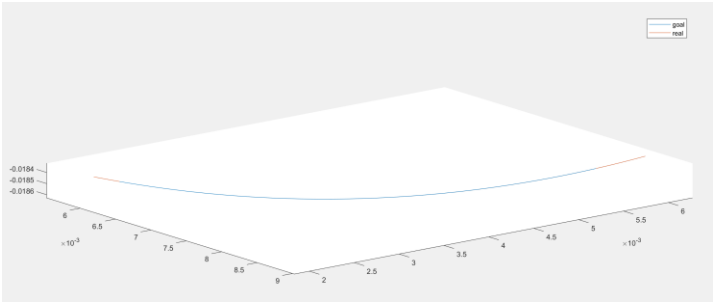
Appendix

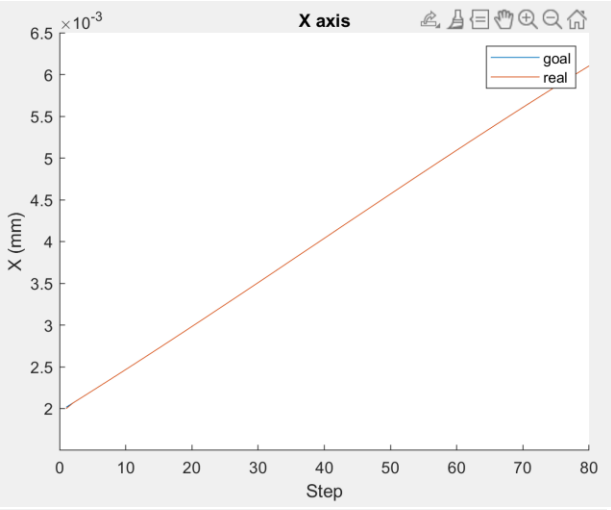
A

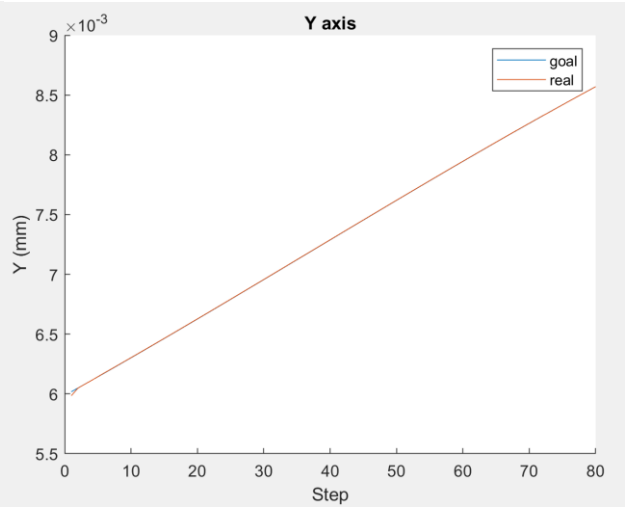

Code Dependency Graph v1
InvKinControlOmni.m
GazeboSub.m
generate_trajectory.m
joint_state_pub.m
Forces.m
q_limits.m
getConstrainedGoal.m
queue.m
InvKinSolver.m
getNewGoalOmni.m
OmniSub.m
q7_joint.m
getSphereNearestNormal.m
Jacobians.m
dq_limits.m
Eye_Robot_q.m
EyeOrigin_RCM.m
generateEyeJacobian.m
q_eye_limits.m
Robot_Eye_q.m
RCM_comp.m
FwdKin_RodTip_SnakeTip.m
FwdKin_EyeOrigin_RodTip.m

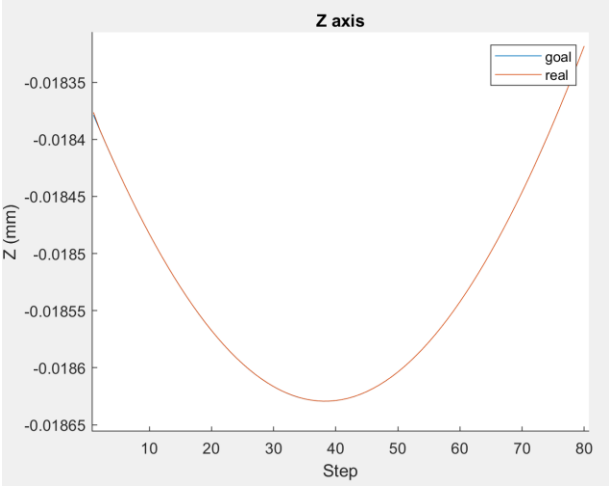
B

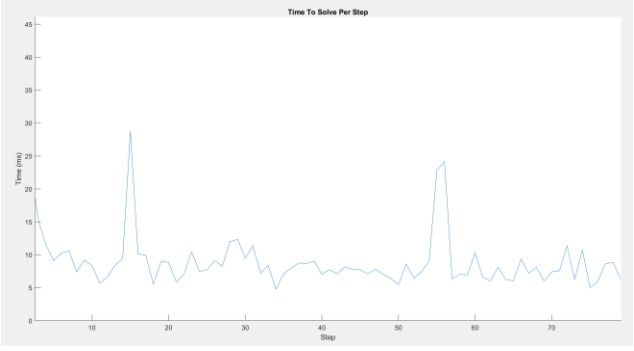
Curved Path











Straight Path