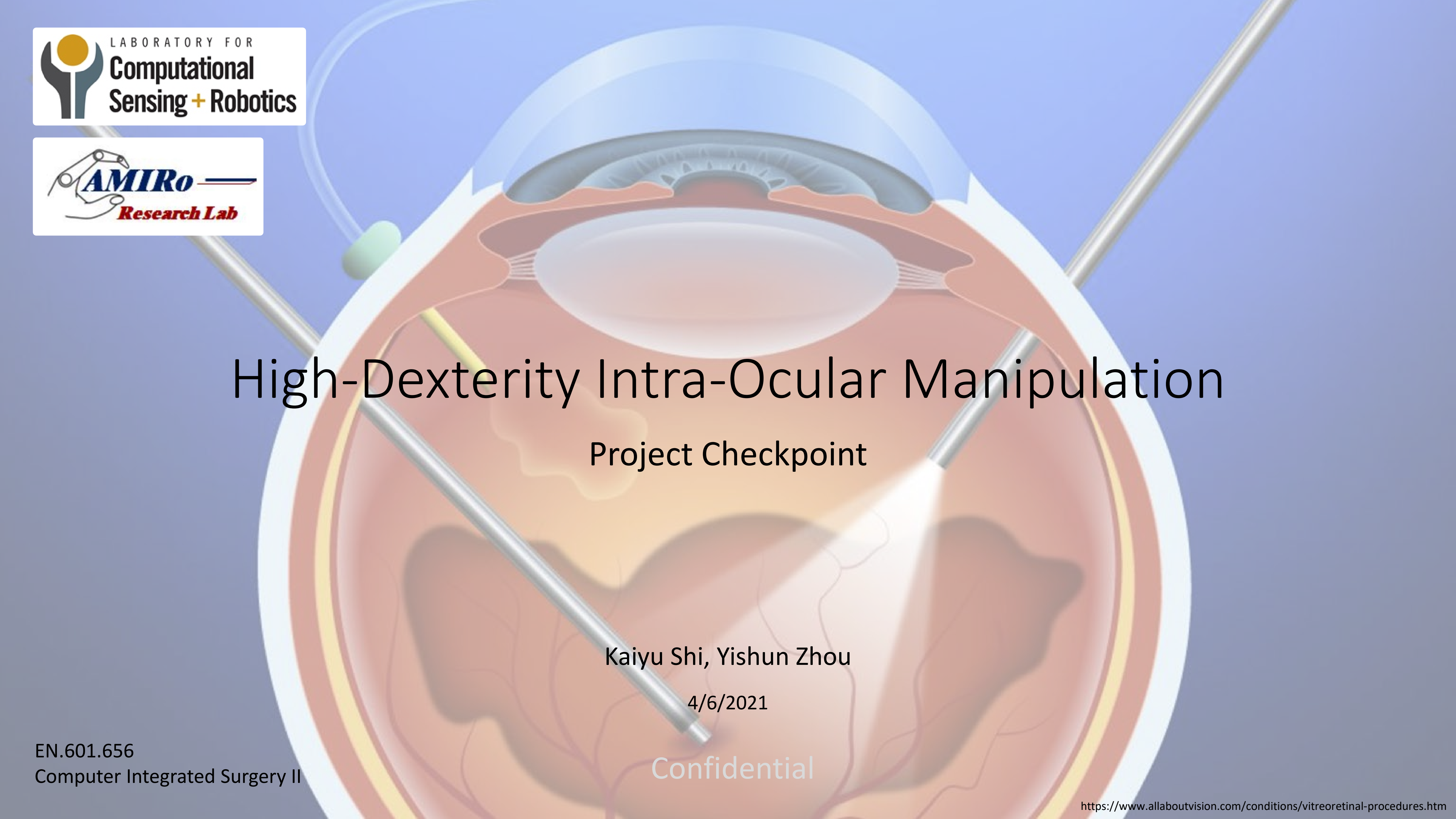


https://www.allaboutvision.com/conditions/vitreoretinal-procedures.htm

High-Dexterity Intra-Ocular Manipulation

Project Checkpoint

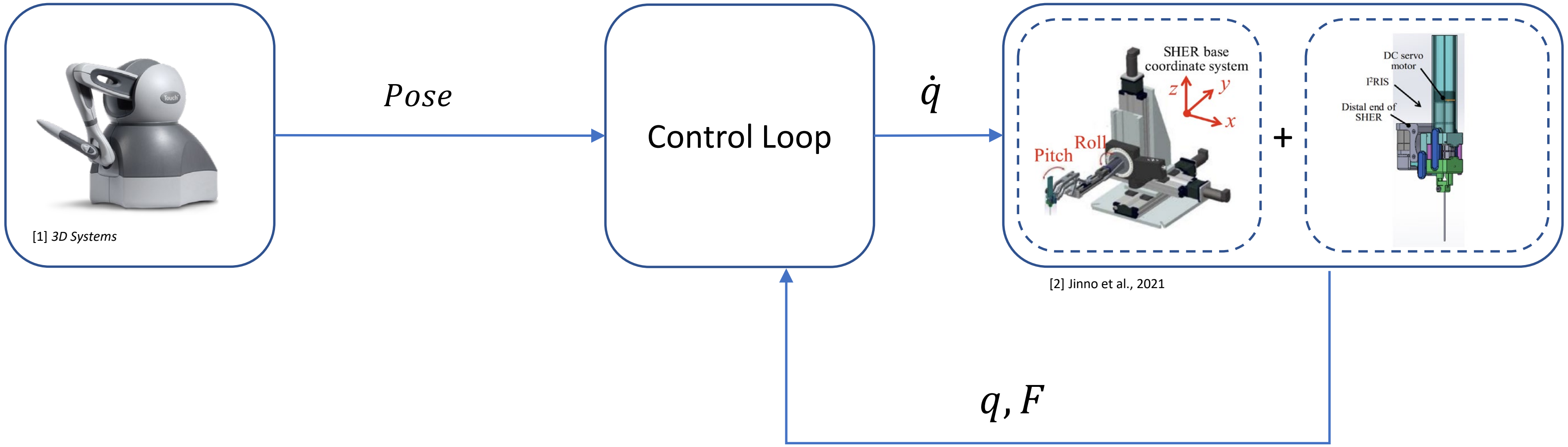
Kaiyu Shi, Yishun Zhou

4/6/2021

EN.601.656
Computer Integrated Surgery II

Project Overview

- Integrate kinematics of 2 DoF distal-end “snake” manipulator with 5 DoF Steady Hand Eye Robot
- Constrained control with Phantom Omni
- Perform simulation with Gazebo, with force sensors



[1] *3D Systems*

[2] Jinno et al., 2021

All images and figures original unless otherwise specified

Dependencies

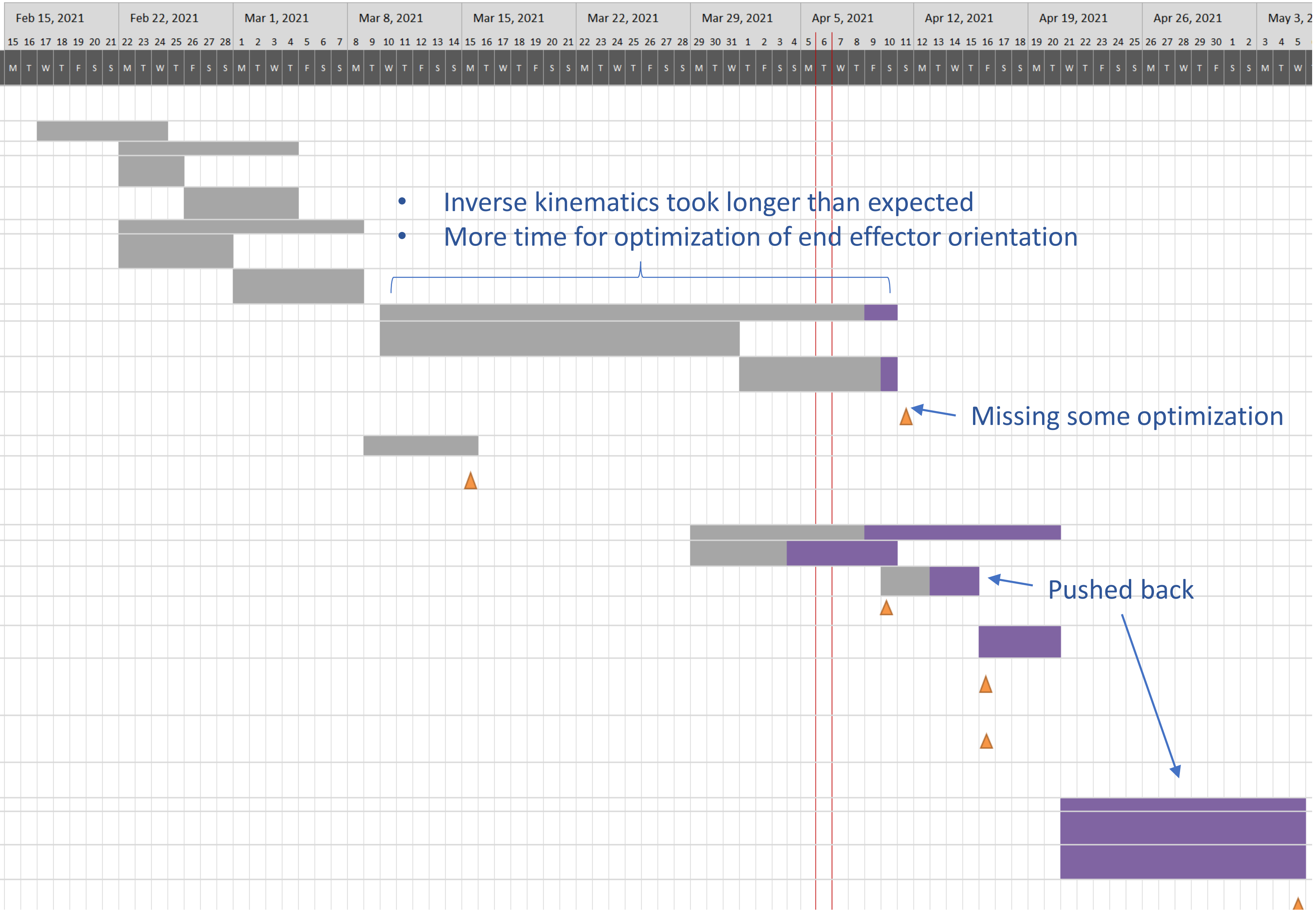
Dependency	Status	Contingency	Followup	Funding	Deadline	Situation	
SHER	Exists	Simulation	-	JHU Internal Funding	-		✓
I^2RIS	No FBG force sensors	Only implement position control/FBG in simulation	Discuss with Prof. Iordachita	JHU Internal Funding	3/29	No FBG sensors	✗
Computer running Linux for simulation	Exists	-	-	Personal computer	-		✓
Phantom Omni	In lab	Joy-stick input/keyboard input	-	JHU Internal Funding	-	Acquired w/ ROS packages & linux laptop	✓

Deliverables

	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	Missing some optimization	A report that includes of inverse kinematics, and jacobian of the combined system	~~3/8~~ 4/10	Missing some optimization
	Force distribution model	3/15	✓	A report on the force distribution analysis	3/15	✓
	Control algorithm pseudocode	~~3/29~~ 4/10	In progress	A schematic of the control algorithm design	~~3/29~~ 4/10	In progress
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	In progress
	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~~ 4/30		Implemented control system on real hardware	5/5	
				Documentation of implementation	5/5	

Timeline

TASK	ASSIGNED	PROGRESS	START	END
Minimum				
Literature review		100%	2/17/21	2/24/21
Setup Gazebo		100%	2/22/21	3/4/21
Setup Gazebo: generate files		100%	2/22/21	2/25/21
Setup Gazebo: create simulation		100%	2/26/21	3/4/21
Forward kinematic model		100%	2/22/21	3/8/21
Forward kinematic model: analysis		100%	2/22/21	2/28/21
Forward kinematic model: coding		100%	3/1/21	3/8/21
inverse kinematic model		95%	3/10/21	4/10/21
Inverse kinematic model: analysis		100%	3/10/21	3/31/21
Inverse kinematic model: coding		95%	4/1/21	4/10/21
A report that includes calculation of forward kinematics, inverse kinematics, and jacobian of the combined system				
Analyze force distribution		100%	3/9/21	3/15/21
A report on the force distribution analysis				
Expected				
Control algorithm		50%	3/29/21	4/20/21
Control algorithm: create psuedocode		50%	3/29/21	4/10/21
Control algorithm: write code		50%	4/10/21	4/15/21
A schematic of the control algorithm design				
Control system: testing and validation		0%	4/16/21	4/20/21
A functioning gazebo simulation in which the end-effector of the simulated eye robot follow several optimized trajectories				
A report summarizing the control algorithm and evaluation of the simulated system				
Maximum				
Real system		0%	4/21/21	5/5/21
Real system: imprement code		0%	4/21/21	5/5/21
Real system: testing and validtion		0%	4/21/21	5/5/21
Documentation of implementation				



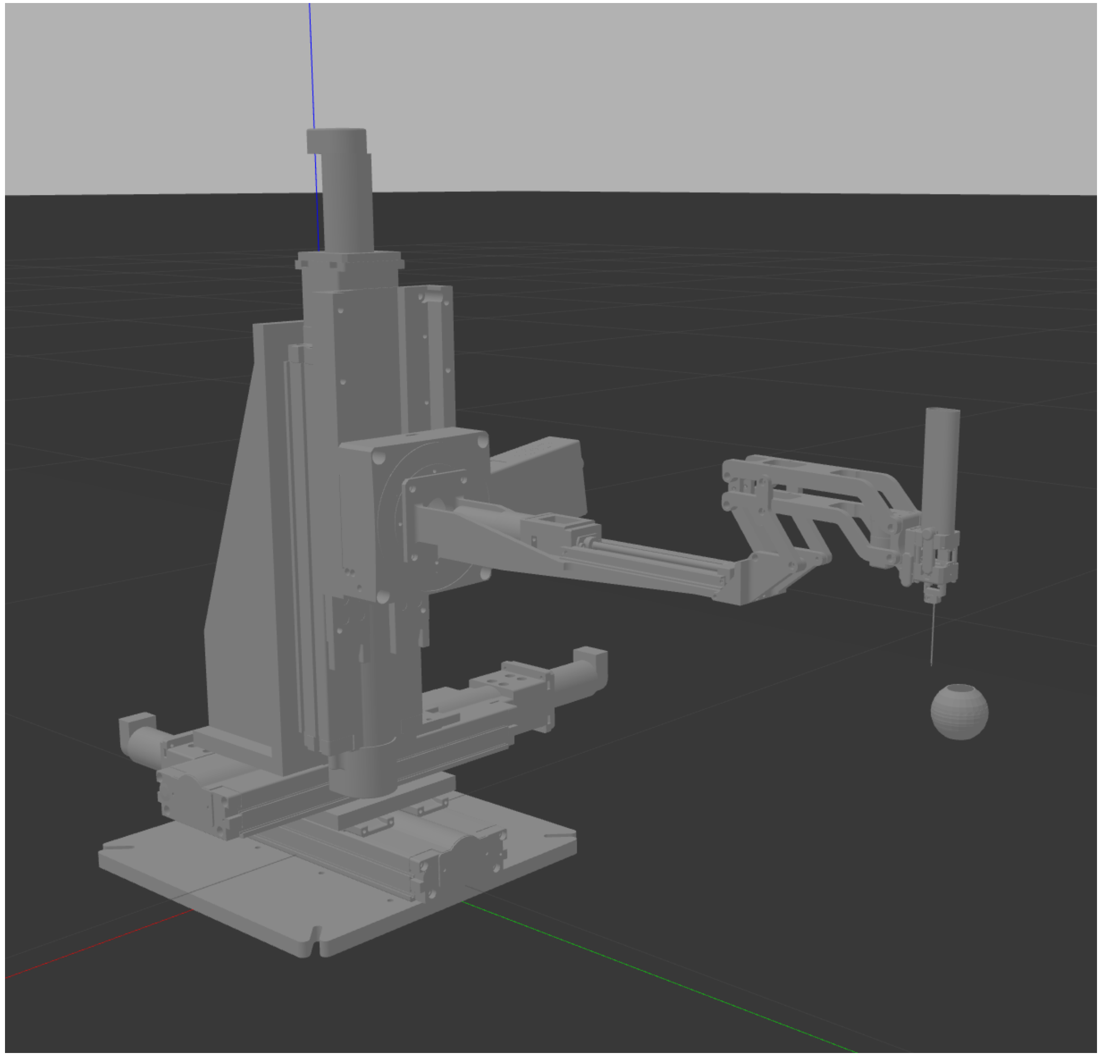
- Inverse kinematics took longer than expected
- More time for optimization of end effector orientation

Missing some optimization

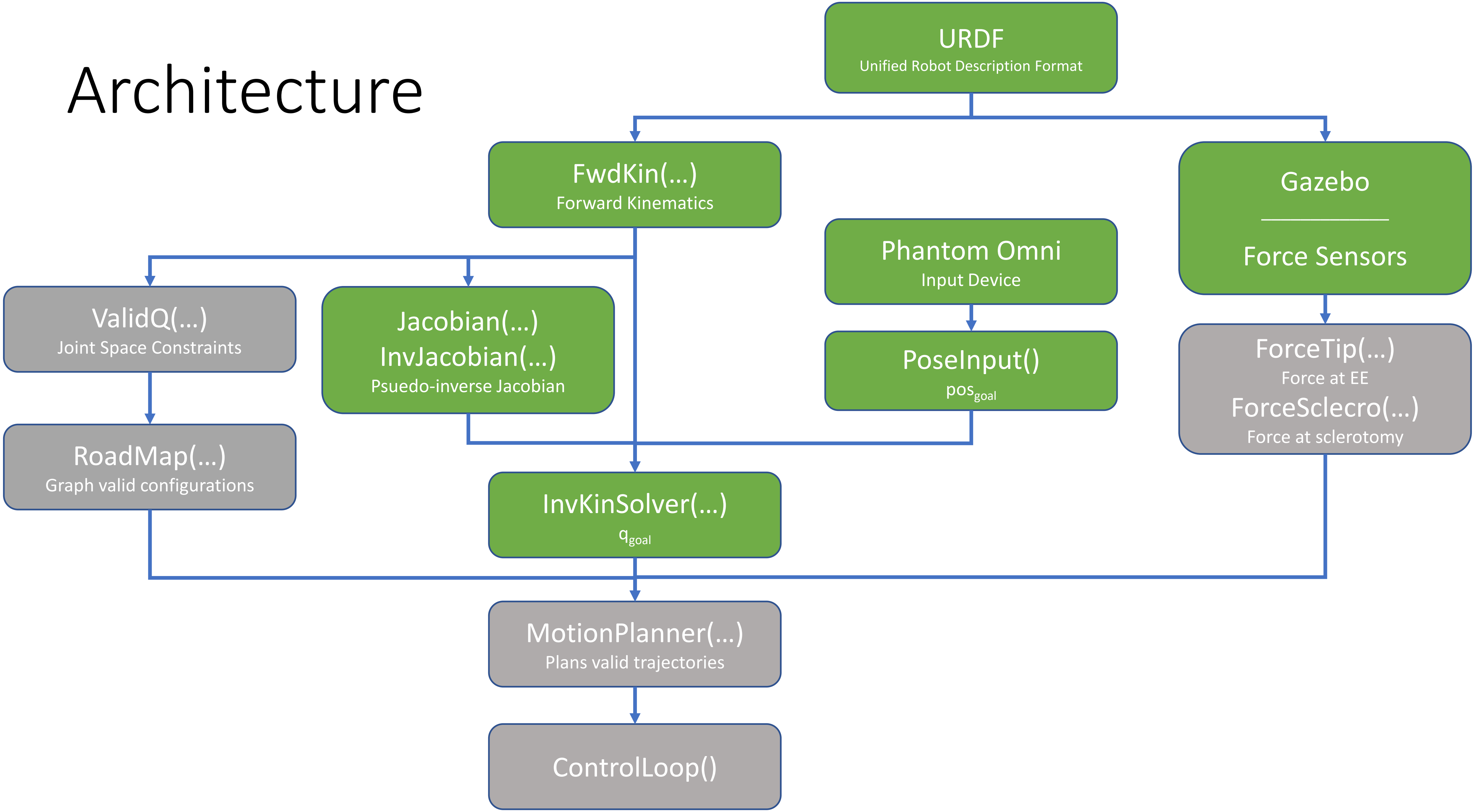
Pushed back

Simulation

- Simulated in Gazebo
- Robot defined in URDF format
- Velocity controlled joints controllable via ROS topic



Architecture



URDF
Unified Robot Description Format

FwdKin(…)
Forward Kinematics

Gazebo

Force Sensors

Phantom Omni
Input Device

ValidQ(…)
Joint Space Constraints

Jacobian(…)
InvJacobian(…)
Psuedo-inverse Jacobian

PoseInput()
pos_{goal}

ForceTip(…)
Force at EE
ForceSclecro(…)
Force at sclerotomy

RoadMap(…)
Graph valid configurations

InvKinSolver(…)
q_{goal}

MotionPlanner(…)
Plans valid trajectories

ControlLoop()

Forward Kinematics [1/3]

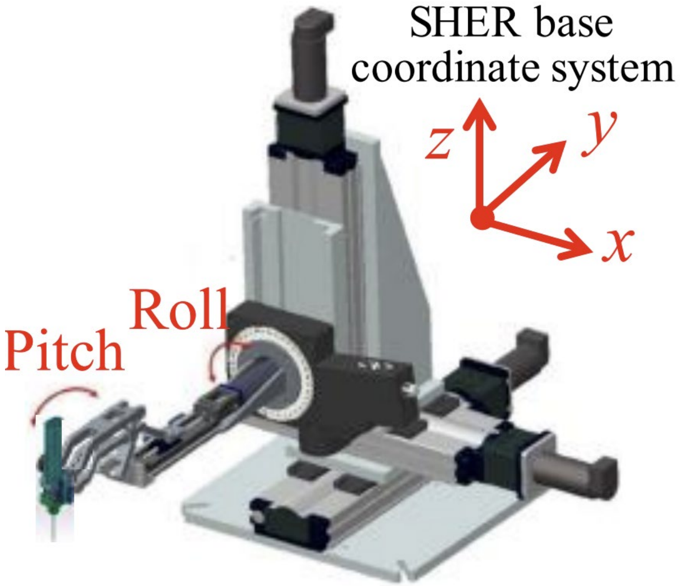
Forward Kinematics of SHER

- q_1: x translation
- q_2: y translation
- q_3: z translation
- q_4: roll
- q_5: pitch

CIS I Transformations:

$F = [\ R\ ,\ p\ ]$

$F_1F_2 = [\ R_1R_2\ ,\ p_1+R_1p_2\ ]$



[2] Jinno et al., 2021

Forward Kinematics [2/3]

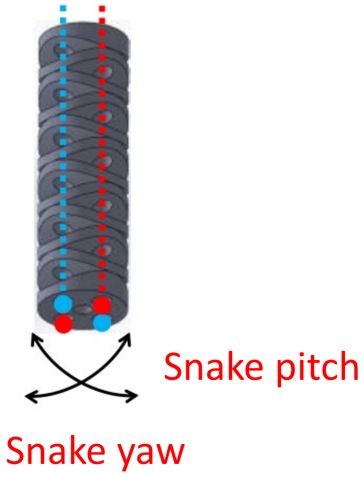
Forward Kinematics of I^{2}RIS (Snake)

- q_6: snake yaw
  - R_6 = Rot((0 1 0),q_6)
- q_7: snake pitch
  - R_7 = Rot((1 0 0),q_7)

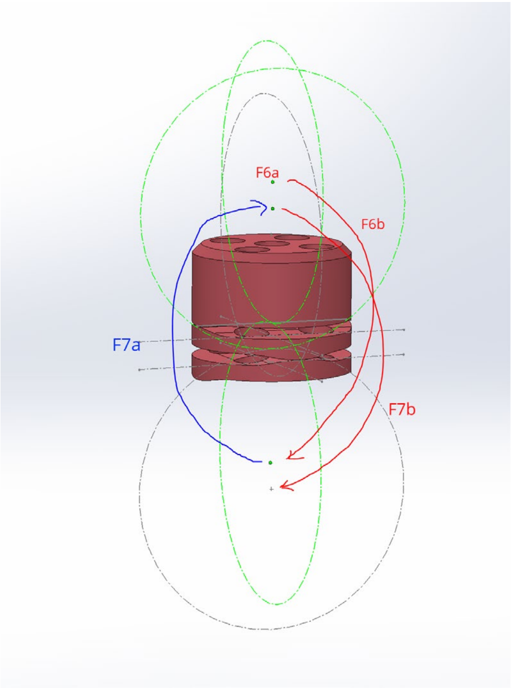
Rolling surfaces complicates kinematics

Solution: pair of virtual joints between each link:

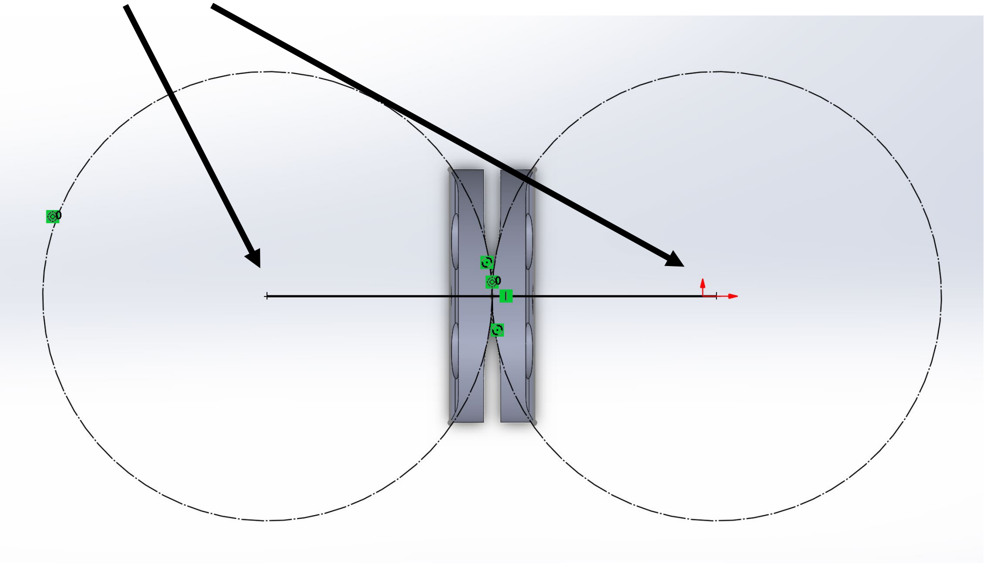
- F_{7a} = [R_7,(0 0 0.00145)]
- F_{7b} = [R_7,(0 0 -0.0016)]
- 12 pairs of such transformation matrices multiplied



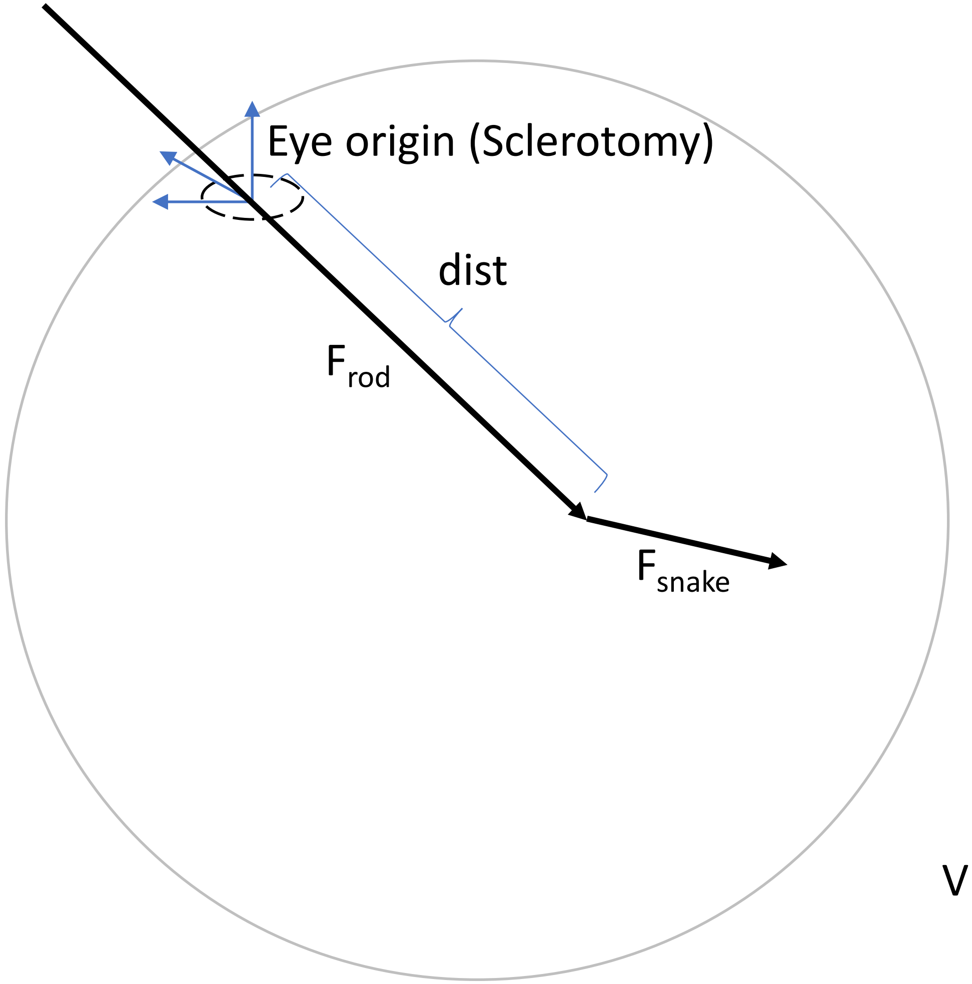
[2] Jinno et al., 2021



Virtual Joints



Forward Kinematics [3/3]

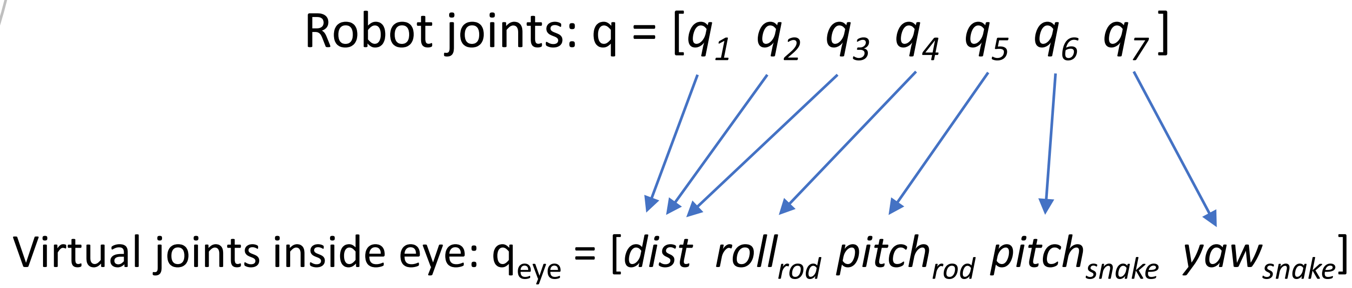


Forward Kinematics from Eye Origin to the tip of Snake

The eye origin frame is defined to be the same orientation as the base of SHER

$F_{rod} = f(\text{dist}, \text{roll}_{rod}, \text{pitch}_{rod})$

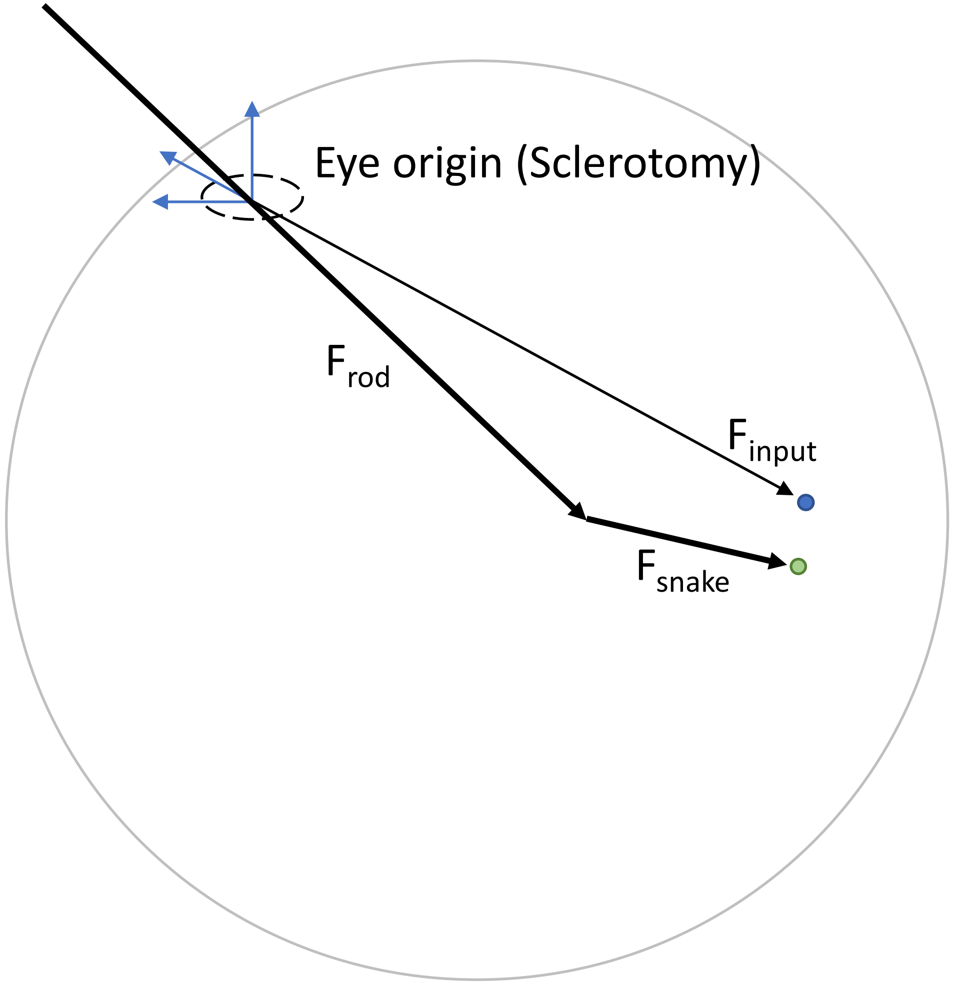
$F_{snake} = f(\text{pitch}_{snake}, \text{yaw}_{snake})$



Robot joints: $q = [q_1\ \ q_2\ \ q_3\ \ q_4\ \ q_5\ \ q_6\ \ q_7]$

Virtual joints inside eye: $q_{eye} = [\mathit{dist}\ \ \mathit{roll}_{rod}\ \mathit{pitch}_{rod}\ \mathit{pitch}_{snake}\ \ \mathit{yaw}_{snake}]$

Inverse Kinematics [1/2]



Phantom Omni input by user is relative to the eye
Therefore, solving inverse kinematics relative to the eye is desirable

$F_{rod} = f(\text{dist},\text{roll}_{rod},\text{pitch}_{rod})$
$F_{snake} = f(\text{pitch}_{snake}, \text{yaw}_{snake})$
$F_{eye} = F_{rod}F_{snake}$

$\delta(F_{eye})/ \delta(q_{eye})$ -> Jacobian

We use gradient descent to find a solution:

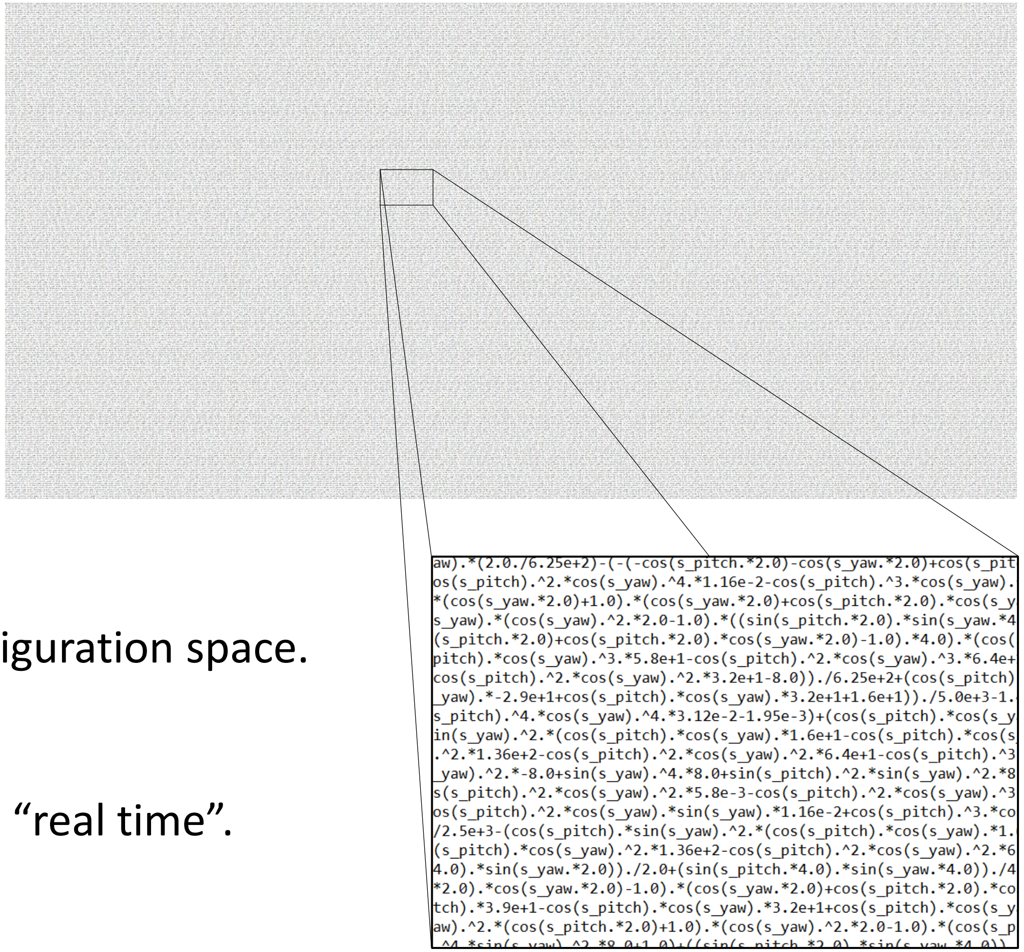
```
while $||\Delta x|| < \varepsilon$:
    psuedoinv(Jacobian($q_{curr}$)) -> InvJacobian
    InvJacobian*($\alpha\Delta x$) -> $\Delta q$
    $q_{curr} + \Delta q$ -> $q_{curr}$
    $q_{goal} – F(q_{curr})$ -> $\Delta x$
return $q_{curr}$
```

Inverse Kinematics [2/2]

Jacobian as a function has a lot of terms:
- Takes ~0.2 seconds to compute in MATLAB
- InverseKinematicsSolver() takes ~2 seconds to find solution

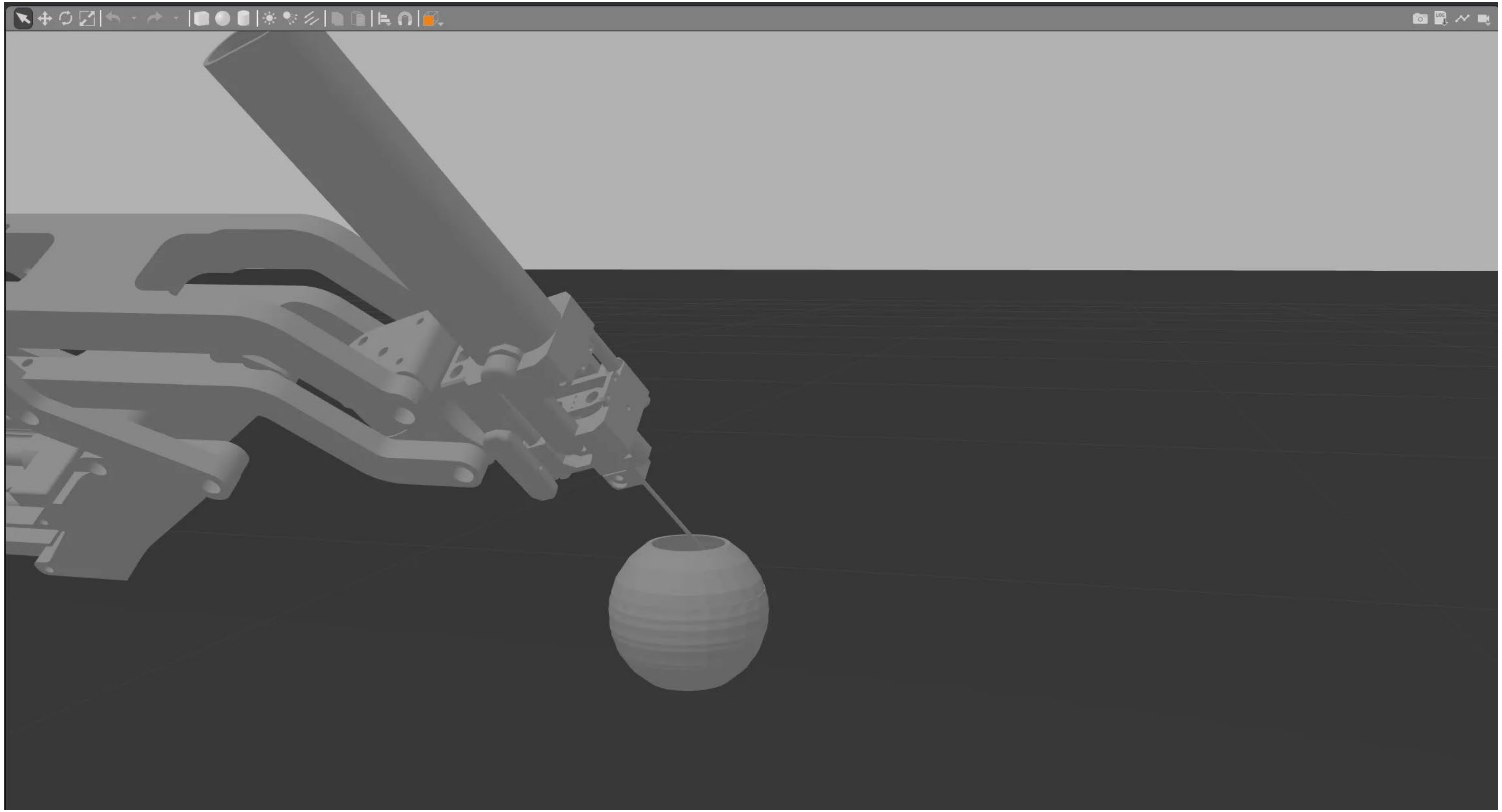
… too slow!

Alternatively, look up table of precalculated inverse jacobians:
- 29 increments for every eye joint, covering full range of configuration space.
- 1.3 GB array loaded once into memory
- Takes ~0.0002 seconds to compute in MATLAB
- InverseKinematicsSolver() takes ~0.002 seconds, can work in “real time”.



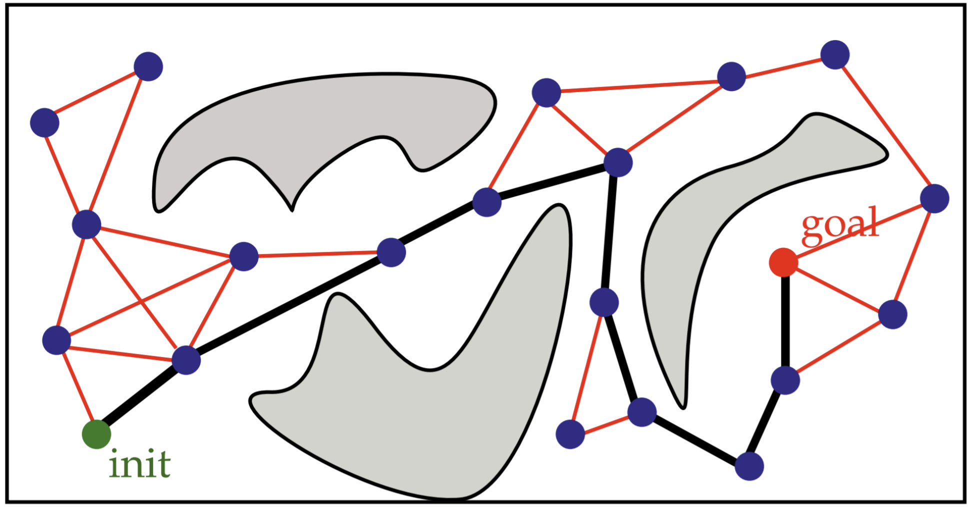
```
aw).*(2.0./6.25e+2)-(-(-cos(s_pitch.*2.0)-cos(s_yaw.*2.0)+cos(s_pit
os(s_pitch).^2.*cos(s_yaw).^4.*1.16e-2-cos(s_pitch).^3.*cos(s_yaw).
*(cos(s_yaw.*2.0)+1.0).*(cos(s_yaw.*2.0)+cos(s_pitch.*2.0).*cos(s_y
s_yaw).*(cos(s_yaw).^2.*2.0-1.0).*((sin(s_pitch.*2.0).*sin(s_yaw.*4
(s_pitch.*2.0)+cos(s_pitch.*2.0).*cos(s_yaw.*2.0)-1.0).*4.0).*(cos(
pitch).*cos(s_yaw).^3.*5.8e+1-cos(s_pitch).^2.*cos(s_yaw).^3.*6.4e+
cos(s_pitch).^2.*cos(s_yaw).^2.*3.2e+1-8.0))./6.25e+2+(cos(s_pitch)
_yaw).*-2.9e+1+cos(s_pitch).*cos(s_yaw).*3.2e+1+1.6e+1))./5.0e+3-1.
s_pitch).^4.*cos(s_yaw).^4.*3.12e-2-1.95e-3)+(cos(s_pitch).*cos(s_y
in(s_yaw).^2.*(cos(s_pitch).*cos(s_yaw).*1.6e+1-cos(s_pitch).*cos(s
.^2.*1.36e+2-cos(s_pitch).^2.*cos(s_yaw).^2.*6.4e+1-cos(s_pitch).^3
_yaw).^2.*-8.0+sin(s_yaw).^4.*8.0+sin(s_pitch).^2.*sin(s_yaw).^2.*8
s(s_pitch).^2.*cos(s_yaw).^2.*5.8e-3-cos(s_pitch).^2.*cos(s_yaw).^3
os(s_pitch).^2.*cos(s_yaw).*sin(s_yaw).*1.16e-2+cos(s_pitch).^3.*co
/2.5e+3-(cos(s_pitch).*sin(s_yaw).^2.*(cos(s_pitch).*cos(s_yaw).*1.
(s_pitch).*cos(s_yaw).^2.*1.36e+2-cos(s_pitch).^2.*cos(s_yaw).^2.*6
4.0).*sin(s_yaw.*2.0))./2.0+(sin(s_pitch.*4.0).*sin(s_yaw.*4.0))./4
*2.0).*cos(s_yaw.*2.0)-1.0).*(cos(s_yaw.*2.0)+cos(s_pitch.*2.0).*co
tch).*3.9e+1-cos(s_pitch).*cos(s_yaw).*3.2e+1+cos(s_pitch).*cos(s_y
aw).^2.*(cos(s_pitch.*2.0)+1.0).*(cos(s_yaw).^2.*2.0-1.0).*(cos(s_p
.^4.*sin(s_yaw).^2.*8.0+1.0)+((sin(s_pitch.*2.0).*sin(s_yaw.*4.0))
```

Moving in Gazebo



Motion Planning

ValidQ() function determines if configurations of q are invalid:
- If end effector is outside ocular workspace
- If configuration intersects or obstructs light emitter
- Other cases

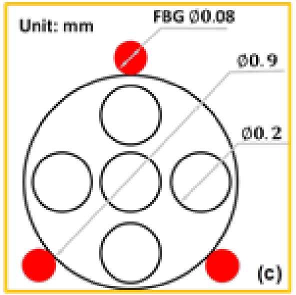


S. LaVelle, E Plaku

1st Method: Linear interpolation
- Generate nodes linearly between q_{curr} and q_{goal}
  - Check every node with ValidQ()

2nd Method: Probabilistic graph search
- Probabilistic RoadMap Planning (PRM) [3]
- Valid path is found by Dijkstra’s algorithm or similar method

Force Distribution Model [1/2]



From Dr. Iordachita

Force at the Tip

- FBG sensor readings can be related to force

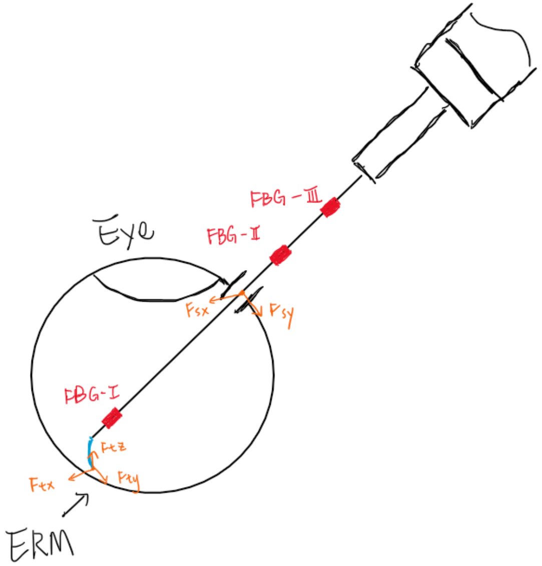
$$\Delta S_I = K_I F_I, \;\; F_I = [F_{Ix}, F_{Iy}]^T$$

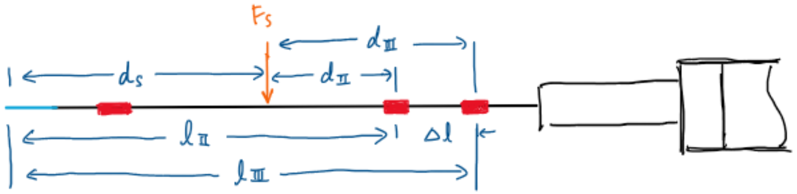
- Solve a system of three equations with three unknowns

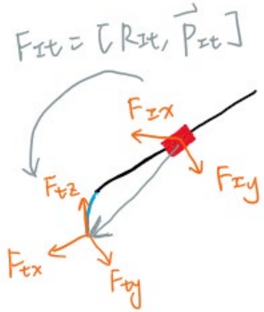
$$\vec{F_t^I} \cdot \vec{x_I} = (R_{It} \cdot \vec{F_t}) \cdot \vec{x_I} = F_{Ix}$$

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

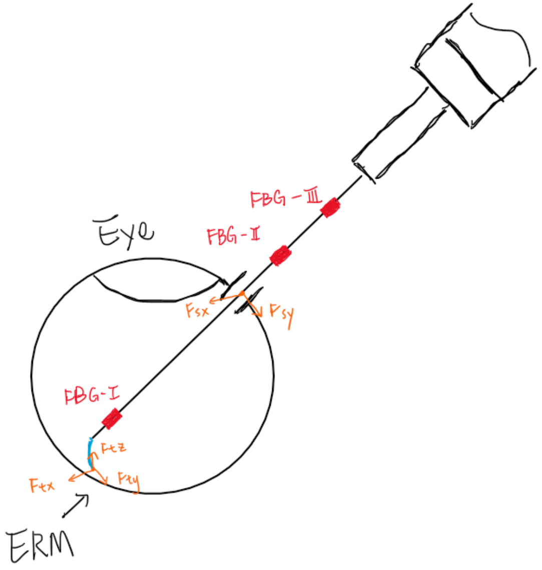
$$(\vec{p}_{It} \times \vec{F_t^I})_3 = (\vec{p}_{It} \times (R_{It} \cdot \vec{F_t}))_3 = 0$$

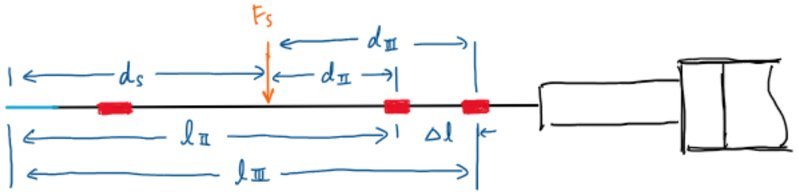






Force Distribution Model [2/2]





Force at the Sclerotomy

- FBG sensor readings can be related to torque

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

- Find out the torque contributed by forces at the tip

$$\vec{\tau}_j = \vec{\tau}_t^{\,j} + \vec{\tau}_s^{\,j}, \ j = II, III$$

$$\vec{\tau}_t^{\,j} = \vec{p}_{jt} \times (R_{jt} \cdot \vec{F}_t)$$

- Solve a system of three equations with three unknowns

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

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

$$d_j = \frac{||\tau_s^j||}{||F_s||}$$

Management Plan

- Meetings:
  - Meet weekly (Wed 11 am) with Dr. Li and Prof. Iordachita over Zoom
  - Meet with Dr. Li in lab as needed
  - Weekly team meetings (Tuesday 4:00-5:00 pm), and on-demand
- Files:
  - Code & CAD: Private Github Repo
  - Literature & Deliverables: OneDrive
- Communications:
  - Email between mentors and the team
  - Slack between the team members

Reference

[1] 3D Systems. “Touch.” *3D Systems*, 4 June 2020, www.3dsystems.com/haptics-devices/touch.
[2] Makoto Jinno, Gang Li, Niravkumar Patel, Iulian Iordachita, “An Integrated High-dexterity Cooperative Robotic Assistant for Intraocular Micromanipulation”, 2021., Kokushikan University
[3] Kavraki, L.E., et al. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces.” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, 1996, pp. 566–580., doi:10.1109/70.508439.

Thank You