Final Report

Autonomous Reaming for Total Hip Replacement

HIPSTER | ARTHuR

Team C:

Kaushik Balasundar | Parker Hill | Anthony Kyu
Sundaram Seivur | Gunjan Sethi

December 14th, 2022

Carnegie Mellon University
The Robotics Institute

Abstract

One of the most crucial factors in performing a successful Total Hip Arthroplasty (THA) surgery is the accuracy of the placement of the acetabular cup within the acetabulum, as placement with the correct position and orientation decreases the likelihood of future dislocations of the hip joint and increases patient comfort. Unfortunately, this is a difficult task for surgeons to undertake and less than 50% of THA surgeries are within surgical safe zones. Thankfully, autonomous robotic technologies could provide a way to execute more patient-specific surgical plans with high accuracy. ARTHuR (Autonomous Reaming for Total Hip Replacement Robot) is a system that will maximize the accuracy of acetabular reaming while remaining robust during an autonomous operation. Using an Atracsys camera, fiducial markers, a Kinova Gen-3 robotic arm, and a custom reaming end-effector, our team has developed a system that can localize a pelvis to a robotic arm and control the arm autonomously to ream a pelvis to a patient-specific surgical plan. This system also makes advancements on similar platforms through its ability to dynamically compensate for movements in the pelvis that may occur naturally during a reaming operation. We demonstrated that ARTHuR was able to ream bone consistently within 3.4 mm of error with respect to the surgical plan.

Contents

1 Project Description

Total Hip Arthroplasty (THA) Surgery, today, involves many steps, including cutting the femoral head, drilling into the femur, placing the femoral stem into the femur, reaming the acetabulum, and placing an acetabular cup into the reamed acetabulum. One of the most crucial factors in determining a successful surgery is the accuracy of acetabular cup position and orientation, which would decrease the likelihood of future dislocation of the hip joint and increase patient comfort [1]. Therefore, it is imperative that surgeons know exactly what depth they are reaming the acetabulum to and what orientation the acetabular cup is placed at.

However, most surgeons cannot see the site of surgery well during surgery and do not use the proper tools to obtain accurate results, leading to malpositioned cups. In fact, it is estimated that less than 50% of THA outcomes are within surgical safe zones (such as the Lewinnek Safe Zone, a constraint that increases surgery success) [2]. This is a result of "intraoperative pelvic tilt, distorted anatomical landmarks, and limited accuracy and reproducibility of the alignment guides." Other factors of malpositioned cups are "minimally invasive surgical approaches, low surgeon volume, and obesity" [1]. While there are modern robotic systems that can help mitigate this problem, all of them increase the time the surgery takes and lack robustness.

"More recently, patient-specific safe zones based on preoperative assessments of pelvic kinematics have gathered momentum as a route for improving stability and reducing complications in THA" [3]. With manual surgery, these plans would be hard to execute with high accuracy. However, autonomous robotic technology can provide a way to execute patient-specific surgical plans to meet these patient-specific safe zones with high accuracy [1]. To meet this need, our team is proposing ARTHuR (Autonomous Reaming for Total Hip Replacement Robot). ARTHuR is a fully autonomous robotic arm aimed at performing acetabular reaming with high accuracy, eliminating the need of surgeons to use intuition to correctly position/angle the reamer.

2 Use Case

2.1 Use Case Narrative

Dr. Williams is an orthopedic surgeon and needs to perform a total hip replacement/arthroplasty on a patient suffering from osteoarthritis in their hip joint. Prior to the beginning of the surgery, Dr. Williams takes a CT scan of the patient's pelvis in order to reconstruct the 3-dimensional geometry of the patient's pelvis, specifically their acetabulum. With this 3D geometry, Dr. Williams is able to use his intuition to choose an acetabular cup specifically suited for the patient, and using a 3-dimensional surgical planner, plan the exact location and orientation of the acetabular cup, such that the resulting prosthesis will be in the Lewinnek Safe Zone.

Dr. Williams then begins the surgery by orienting the patient to their back such that the hip which is being operated on is facing sideways for an anterior approach. Dr. Williams, using the typical surgical techniques for hip replacement surgeries, cuts the femoral stem to expose the acetabulum and cleans the desired amount of soft tissue from the pelvis. With the acetabulum exposed, the surgeon then drills a reflective marker array into the pelvis' Iliac Crest which is used

to locate the pelvis in 3-dimensional space with a tracking camera. Using a registration probe with a reflective marker array attached to it, the surgeon registers the position of 3-5 points of interest on the pelvis and then generates a point cloud that represents the anatomy of the patient's pelvis. This point cloud and the points of interest are then used to localize the hip in 3-dimensional space, fitting the 3D geometry from the CT scan to the patient's pelvis on the operating table. Given this localization and the desired acetabular cup location (from a surgical plan), a reaming end-pose is then generated which a robotic arm will follow in order to ream the acetabulum as desired. Dr. Williams then fixes a reamer onto the end-effector of the robotic arm. From here, the doctor manipulates the robotic arm such that the end-effector is near where the reamer would begin cutting into the acetabulum. Dr. Williams then starts the robotic arm using the surgeon's user interface, which begins to autonomously track the reaming end-pose and ream the acetabulum according to the surgical plan. During the reaming, the robot arm detects and compensates for movements in the patient's pelvis that occur as a result of the applied forces, maintaining the surgical plan. Furthermore, Dr. Williams is provided with visual feedback on a monitor, which demonstrates the current progress in reaming the acetabulum. He also has access to an emergency stop button.

Once the robot arm has reached the reaming end-pose, it stops and allows Dr. Williams to remove it from the surgical site so they can analyze the resulting reamed acetabulum. After the reaming operation is deemed a success, Dr. Williams can then proceed with the rest of the operation by first hammering the selected acetabular cup into the reamed acetabulum. Dr. Williams then cuts into the femur and implants the femoral stem of the hip implant into it, before inserting the femoral head into the acetabular cup to complete the prosthetic hip joint. With the femoral stem and acetabular cup connected, Dr. Williams is free to stitch the patient up and send them on their way home, happy and pain-free.

2.2 Use Case Graphical Representation

Figure 1 shows the implant placement process and the ARTHuR system in its **use case/mission** environment. An enlarged version of this graphic can be found in the appendix.

Figure 1: Use case graphical representation

3 System-Level Requirements

The following requirements were elicited from discussions with our sponsor and with surgeons, while also factoring in the given resources and time.

3.1 Mandatory System-Level Requirements

3.1.1 Mandatory Functional & Performance Requirements

Table 1: Mandatory Functional & Performance Requirements

Functional Requirement	Performance Requirement	Justification
M.F.1 The system shall use the Atracsys camera to track the pelvis, registration probe, and robot arm markers.	**M.P.1.1** The system shall use the Atracsys camera to track the pelvis, registration probe, and robot arm markers with a frame rate greater than or equal to 50 Hz and latency less than or equal to 25 milliseconds.	From Atracsys Sprytrack 300 camera's specifications sheet.
	M.P.1.2 The system shall use the Atracsys camera to track the pelvis, registration probe, and robot arm markers with an accuracy of less than or equal to 0.55 mm.	State-of-the-art, FDA-approved medical tracking systems are able to track fiducial pose with a position accuracy of 0.5 mm.
M.F.2 The system shall continuously calculate the error in pelvis movement.	**M.P.2.1** The system shall continuously calculate the error in pelvis movement with a frame rate greater than or equal to 40 Hz or latency less than or equal to 25 milliseconds.	Values are derived from the specified tracking performance and performance of using Eigen to calculate simple Euclidean distance.
	M.P.2.2 The system shall use the Atracsys camera to track the pelvis, registration probe, and robot arm markers with a positional accuracy less than or equal to 2 mm, and orientation accuracy less than or equal to 1.5 degrees.	
M.F.3 The system shall perform registration between the collected point cloud and the given 3D pelvis scan.	**M.P.3.1** The system shall perform registration between the collected pointcloud and the given 3D pelvis scan with a root mean square (RMS) error of 1 mm.	Time constraints on gathering points with a probe and the inaccuracies in the camera's detection of probe location combine to make it difficult to register a point cloud to a 3D pelvis scan with greater precision.
M.F.4 The system shall dynamically compensate for the movement of the pelvis.	**M.P.4.1** The system shall start dynamically compensating for the movement of the pelvis by commanding the end-effector to retract and/or power off the reamer with a latency of less than or equal to 25 ms when the error thresholds exceed 2 mm and 1.5 degrees.	Based on M.F.2 and M.F.5 the system must not ream while the error is greater than the acceptable thresholds, and must therefore turn off the reamer while realigning with the pelvis.
	M.P.4.2 The system shall dynamically compensate for the movement of the pelvis by beginning to realign the reamer with a latency of less than or equal to 50 ms.	Based on the current controller frequency with improved PID. Kinova Controller API is capped at 40 hz, so it can't be faster than 25 ms.
M.F.5 The system shall ream the pelvis based on the provided surgical plan.	**M.P.5.1** The system shall ream the pelvis based on the provided surgical plan with a positional accuracy of 2 mm.	Based on the extensive literature survey conducted and getting feedback from surgeons and our sponsors, these accuracy values are acceptable within the Lewinnek Safe Zone.
	M.P.5.2 The system shall ream the pelvis based on the provided surgical plan with an orientation accuracy of 1.5 degrees.	
M.F.6 The system shall allow the surgeon to place the robot arm at an initial position	**M.P.6** The system will allow the surgeon to place the robot arm to an initial position by back-driving the robotic arm	Reduce system complexity and prevent the arm from making large motions around the patient and surgeons.
M.F.7 The system shall provide the surgeon with visual feedback	**M.P.7** The system will provide the surgeon with visual feedback with a latency less than or equal to 150 ms	From literature on telesurgery, latency 150 ms is found to be noticeable to surgeons, and degrades the performance of surgeon-performed tasks
M.F.8 The system shall allow the surgeon to e-stop	**M.P.8** The system will allow the surgeon to e-stop the system, stopping the system within 500 ms	Competitor systems have similar quantification

Shown in Table 1, are our mandatory functional and performance requirements. Since the CDRR, many of these requirements have been modified to reflect the system architecture changes. For instance, we no longer have a planning subsystem, so that requirement was removed, and replaced with a requirement on dynamic compensation. In addition, some metrics were tightened and adjusted to reflect the desires of the company sponsor better, such as the error thresholds for executing a surgical plan.

3.1.2 Mandatory Non-Functional Requirements

M.N.1 The system will produce forces low enough for it to be safe around humans.

M.N.2 The system will provide a minimal and easy-to-interpret user interface design for surgeons.

M.N.3 The system will autonomously detect malfunctions and errors and notify the user accordingly.

3.2 Desired System-Level Requirements

3.2.1 Desired Functional & Performance Requirements

Table 2 shows the desired functional and performance requirements.

Table 2: Desired Functional & Performance Requirements

Functional Requirement	Performance Requirement	Justification
D.F.1 The system shall allow surgeon to change end-effector/tool	**D.P.1** The system will allow surgeon to change end-effector/tool in 3 steps or less	During surgery, some surgeons would prefer to step up reamers sizes instead of using the final size. Allowing them to do this in 3 steps or will allow the surgeon to quickly change tools during surgery.
D.F.2 The system shall position acetabular cup for surgeon to impact into reamed acetabulum	**D.P.2** The system will position acetabular cup for surgeon to impact into reamed acetabulum with less than 3-degree error.	According to surgeon surveys and literature review, 3-degree error is the maximum allowable error. Since the reaming determines positional error, positional error is not a concern for impaction.

3.2.2 Desired Non-Functional Requirements

D.N.1 The system will allow for numerous surgeries, without the need for servicing and calibration.

D.N.2 The system will have a cost comparable to similar systems on the market.

D.N.3 The system will adhere to all relevant ISO standards for medical robotic systems.

D.N.4 The system will be of a size/dimension that is ergonomic.

D.N.5 The system will be designed such that it can be serviced easily.

D.N.6 The system will be designed to be easily sterilizable or sterile in the sterile field.

4 Functional Architecture

The functional architecture embedded within a process flow diagram is shown in Figure 2. This structure was adopted to highlight the temporal components and dependencies in our processes effectively. The horizontal axis represents the time at which each process occurs. Broadly, the diagram is divided into three sections that represent the inputs, the system, and the outputs. Their spatial position along the x-axis represents the timing of their occurrence. Two processes placed one below the other indicate that both happen in tandem. The **inputs** are as follows:

1. The pelvis and robot arm poses inferred by their respective marker poses using the tracking camera.
2. The 3D model of the pelvis of the patient obtained pre-operatively, such as using a CT-Scan.
3. 3D model of the robot arm, in the form of a URDF.
4. The surgical plan provided in the form of the pose of the acetabular implant with respect to the 3D model of the pelvis.
5. The registration probe used by the surgeon to match key landmarks of the patient's pelvis to the 3D model obtained pre-operatively.
6. The surgeon's input to initialize the position of the robot arm, and control the e-stop button.

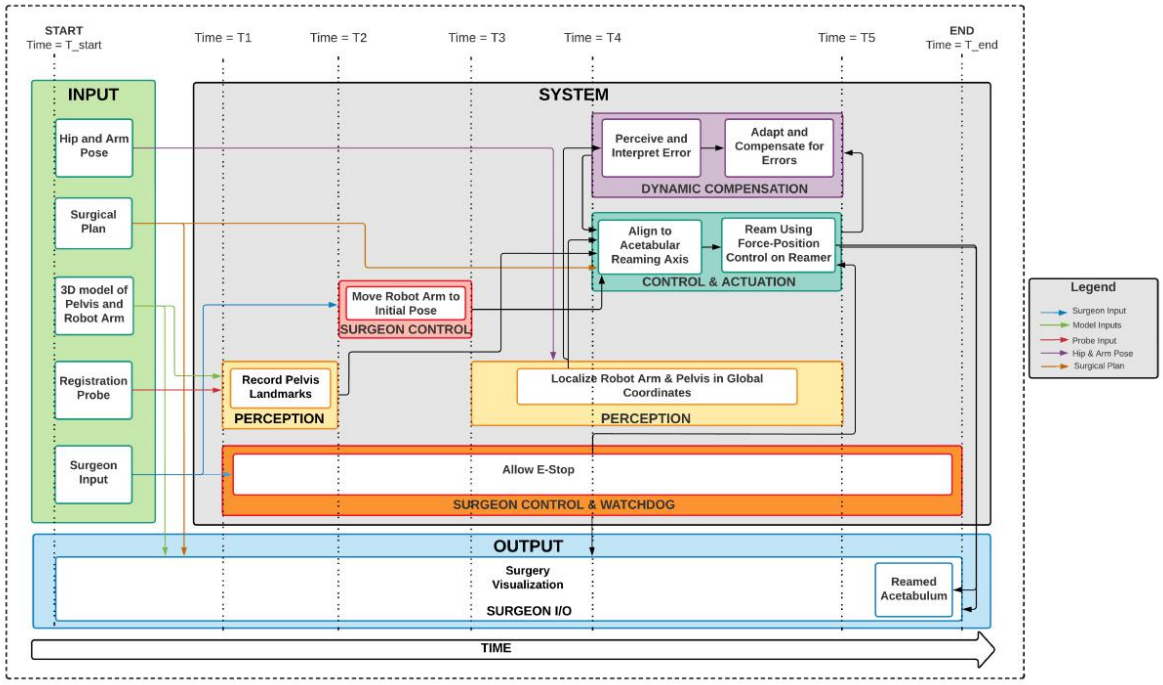

Figure 2: Functional Architecture Embedded in a Process Flow Diagram

In the **system** block, a registration probe is used by the surgeon to record the pelvis landmarks as the first step. Next, the arm is moved by the surgeon to the start position where the acetabulum has been exposed for the reaming process. The arm is localized by the perception system in base-frame coordinates using the markers on the robot arm and the arm's joint encoders. The end-effector marker is used as a calibration target to perform online hand-eye calibration throughout the surgery. The control and actuation as well as the dynamic compensation blocks are placed one below the other, indicating that both processes are happening simultaneously and depend on one another. The dynamic compensation module ensures constant realignment as the pelvis moves from its initial position when impinged upon by the reaming tool. The watchdog is active throughout the process, ensuring that the system is forced into an emergency stop as necessary and alerting the surgeon for manual intervention when something goes wrong.

Finally, the **output** is the reamed acetabulum as per the accuracy thresholds defined in our performance requirements. Throughout this process, the various steps and current status of the surgery can be visualized on a monitor which forms the surgeon I/O component of our system. An enlarged version of this functional architecture can be found in the appendix in Figure 31.

5 System-Level Trade Studies

1. **Investigating Levels of Autonomy:** The goal of robotics and automation in the medical industry is to make medical procedures safer and inexpensive for patients by assisting doctors to perform complex procedures with greater precision, flexibility, and control. In this regard, we conducted a system-level trade study where we compared the varying levels of autonomy: non-robotic hip replacement, computer-guided hip replacement, semi-autonomous robotic hip replacement, and fully-autonomous robotic hip replacement. The primary goal of our system is to improve the accuracy of acetabular reaming in total hip replacement while providing the surgeon with adequate feedback on the status of the procedure. Hence, these criteria were weighted higher in comparison to other criteria such as surgical time, system failure, and effort of setup among others. The study, as seen in Figure 33 in the Appendix, elucidated that a fully-autonomous robot would have the greatest impact on the accuracy of reaming and fared better than the other alternatives.

2. **Choosing an IK Package:** This sub-system level trade study delves into the comparison between the various inverse kinematics packages we could implement in our system. The candidates IKFast, KDL, and TRAC-IK. For the evaluation criteria, the robustness of the solver defined in terms of its average success rate, and the number of unique solutions, form the dominant weighting criteria. Another key element we needed to consider was latency and performance for the dynamic compensation to work effectively. The other two factors are the level of integration with ROS and Moveit!, as well as the documentation and support available for implementation. The candidate that appeared the best was the IKFast plugin. The primary reason was that, unlike the other two alternatives, the IKFast is an analytical solver. It also provides extremely stable solutions that can be found in a few microseconds on recent processors. In the Fall Semester, we decided to write our own IK Package on top of KDL because the candidates of this study were not flexible enough for Task Prioritization. We used KDL as the base package because one of us had extensive experience with KDL

and it provided stable solutions too.

3. **Robot arm choice:** The robot arm is the most critical aspect of our system. We performed a trade study considering the maximum payload, resolution and accuracy, degrees of freedom, ease of integration with other middlewares, etc. Unfortunately, the decision for the choice of the robot arm was out of our control, since we were constrained on using Kinova arms due to our sponsor's relationship with the company. We were initially provided with a Kinova Link 6 robot arm, which had the necessary hardware capabilities. However, it had no ROS API support, which made it unusable for the project. Despite having limitations with its hardware capabilities, we eventually decided to use a Kinova Gen-3 due to its superior API and ROS support.

4. **User-interface Visualization Software:** We had to choose among several options for the user interface and registration software. Qt is a full-fledged GUI development toolkit that allows users to develop fancy user interfaces. VTK and 3DSlicer are toolkits used for the visualization and processing of 3D medical data, and PyVista is a Python binding for VTK that makes it easier to use. However, the most versatile, well-documented, and easy-to-use option that also provided all the necessary features for our project was Open3D, which proved to be the right choice in retrospect.

6 Cyberphysical Architecture

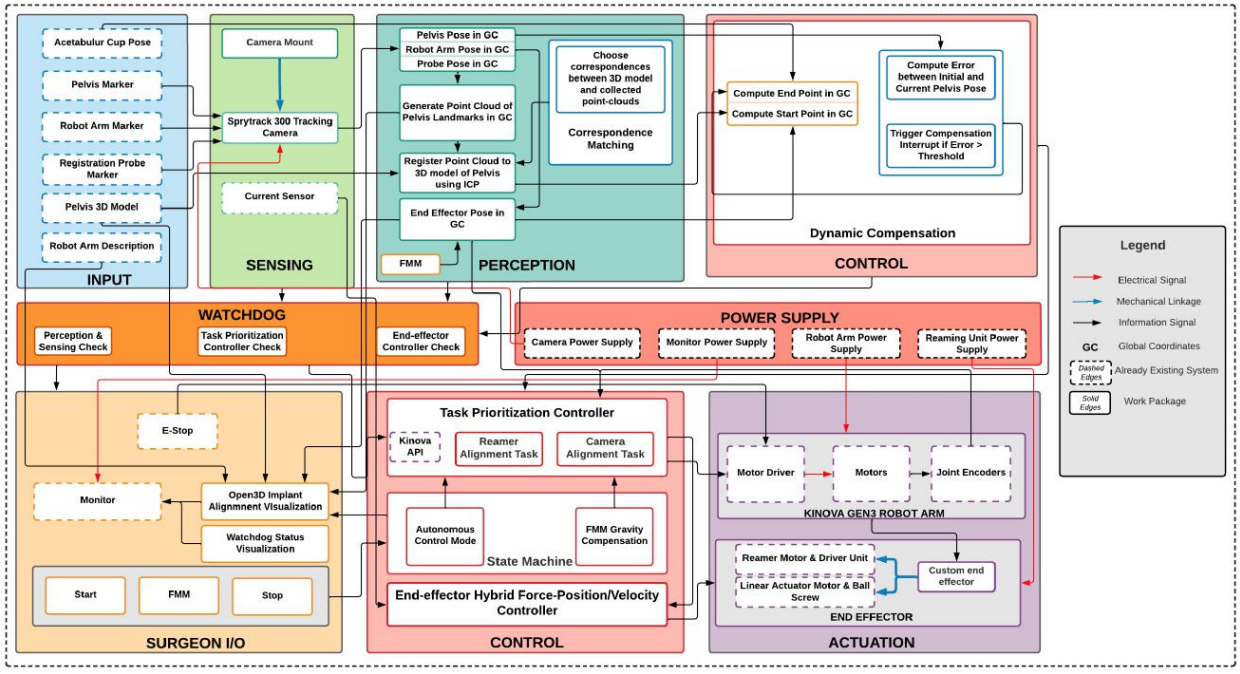

Figure 3: Cyberphysical Architecture

Figure 3 above shows the cyberphysical architecture of our system. The major components of the cyberphysical architecture are derived from the functional architecture. On a high level,

the various blocks are inputs, sensing, perception, control, actuation, watchdog, and surgeon I/O. Blocks having a dashed border indicate the use of off-the-shelf components, and those with a solid border have been developed by the team. The blue arrows indicate mechanical linkages, the black arrows indicate information flow and the red arrows are power flow. An enlarged version of this architecture can be found in the appendix.

The following are brief descriptions of each sub-system involved in the cyberphysical architecture:

1. **Input**: There are six distinct inputs in the input block which have been detailed from the input block in our functional architecture. Three of these are derived from the reflective markers attached to the pelvis, robot arm, and registration probe respectively. The surgical plan in the form of the acetabular cup pose is used to compute the end pose of the reamer. The final inputs are the pelvis and robot arm descriptions. The former is usually obtained from a pre-operative CT scan.

2. **Sensing**: The sensing module is primarily composed of the joint encoders used to compute the joint positions of each joint of the arm. The marker positions are determined using the Atrycsys Sprytrack 300 camera, which is our primary sensing component. This will be securely mounted to externally overlook the arm and the pelvis during the surgery. We have also incorporated an ACS 712 current sensor into our end-effector, which is used to indirectly determine the amount of force being applied by the linear actuator carriage assembly on the acetabulum. We also have limit switches for calibration and over-actuation purposes.

3. **Perception**: Within the perception block, the camera tracks the respective marker geometries to determine the pose of the patient's pelvis, robot arm, and registration marker with respect to the camera. An online calibration routine utilizes the pose of the robot arm marker with respect to the camera to determine the pose of the robot's base link with respect to the camera. The robot arm's configuration is updated at 40 Hz using the robot arm's joint encoders. This process is handled by Kinova's API. A one-time process of landmark acquisition happens at the beginning of each surgery where key landmarks from the patient's pelvis are captured to form a point cloud. This resulting point cloud is then fit to the 3D model of the pelvis obtained pre-operatively. This process localizes the acetabulum with respect to the pelvis marker. The dynamic compensation feature utilizes the pose of the pelvis marker to constantly update the end pose to be tracked by the robot arm's task prioritization controller.

4. **Control**: The controls sub-system is composed of the task prioritization controller and the end-effector hybrid force-position controller. The surgeon will first use the free motion mode to place the robot arm close to the acetabulum. The task prioritization controller utilizes the pose of the pelvis marker and the surgical plan communicated through the user interface, to dynamically track the acetabulum at the desired pose. Once the arm positions itself at the reaming pose, the end-effector controls take over the reaming process to remove bone using the reamer motor. The surgeon can use the user interface to switch to free-motion mode at any point during surgery.

5. **Actuation**: At the heart of our actuation sub-system is our fully custom-designed and manufactured robot end-effector. It consists of a reamer motor that rotates the reamer and a linear

actuator that advances the reamer carriage using a ball screw. They work in conjunction with current sensors which we use for force feedback. The control architecture is a hybrid dual-loop force position controller. A high-level force controller ensures that the reamer applies the right amount of force on the system, and a low-level position or velocity controller executes the output of the force controller accordingly. The heart of the controller is a state machine that transitions between the end-effector's various states during the different stages of the system's state.

6. **Surgeon User Interface**: The user interface integrates with the various subsystems and provides a portal through which the surgeon is able to communicate with the system. Surgeons are able to collect landmark points, match them with the preoperative CT scan, and finally visualize and plan the surgery. Once the surgeon is happy with the implant-pelvis alignment, they are able to communicate this with the system and monitor the autonomous reaming operation. It also displays any system faults from the watchdog that require manual intervention. The surgeon is also able to stop the system and manually intervene at any point during the surgery using this user interface.

7 System Description and Evaluation

7.1 Overall system depiction

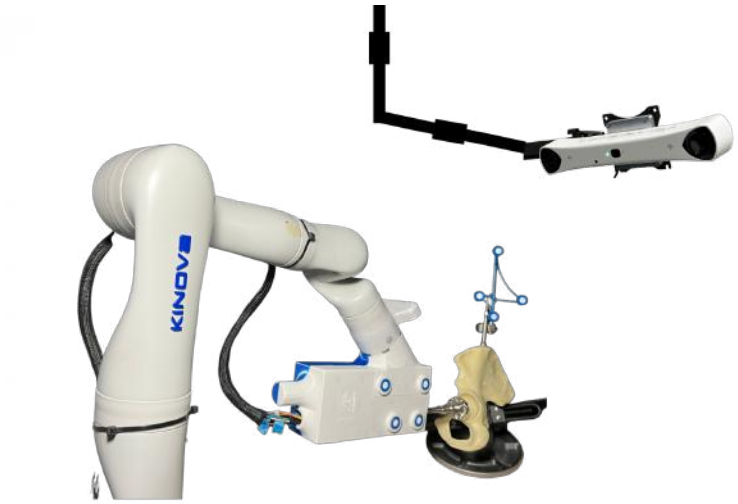

Figure 4: Overall System Depiction

Our system has several integrated software, hardware, and electrical components which detail the full status of the current system. Figure 4 depicts the final status of our system. At the top of the figure is our Atracsys Sprytrack 300 camera, which is composed of two cameras that can track infrared fiducial markers that are attached to specific places in a surgical procedure. This camera is mounted to a VESA monitor mount via a custom camera adapter machined out of aluminum in the RI machine shop. Our robot arm is a Kinova Gen-3 and it was placed towards the back middle

of our workspace providing a good angle to the pelvis. Attached to the end-effector of the Gen-3 arm is our custom reaming end-effector, which is comprised of manufactured aluminum structural components, two DC planetary gear motors with encoders, two limit switches, a ballscrew, a reaming shaft, a 3D printed cover, and fiducial markers for online registration.

Figure 5: Top of Workspace

Figure 5 depicts our workspace during an operation. The base of it is a Vention table which we received from Professor Kroemer. We added wooden boards to the top and bottom of our workspace, one for ease of working with the arm and the surgical setup on top, and one for placing all of our electrical components on the bottom. We encased our pelvis in ballistics gel to simulate the vibrations of the pelvis in soft tissue and used sandbags to secure the pelvis in place during our testing to simulate the weight of a patient. Attached to the pelvis is a fiducial marker to track the pelvis, which is attached to a screw that was drilled into the iliac crest.

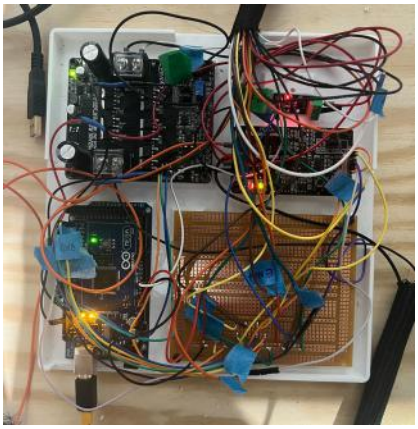

Figure 6: Bottom of Workspace

On the bottom of our workspace, which can be seen in Figure 6, is where we kept all our electrical components. Down there is our Arduino Mega microcontroller, two Cytron motor controllers, a protoboard for connections, an ACS712 current sensor, and a 12V 30A power supply. Not included in any of our images is an integral part of our system, our desktop computer which runs a ROS Noetic workspace and handles all our data processing, planning, and user interfacing.

Connected to the computer are several USB and Ethernet cables that are used for interfacing with the camera, robotic arm, and end effector microcontroller.

7.2 Subsystem descriptions/depictions

7.2.1 Inputs

The inputs into the system and their intended functionality are as follows:

1. Pelvis Marker: A set of fiducial markers that allow the camera to track the pose of the pelvis on the surgical table.
2. End-Effector Marker: A set of fiducial markers that allow the camera to track the pose of the robot arm end-effector. Currently, we are using this as a calibration target for online hand-eye calibration.
3. Registration Probe: A set of fiducial markers mounted on a probe with a 1 mm radius tip that can be used to collect a point cloud of the acetabulum and surrounding iliac crest features for registration before surgery.
4. Pelvis 3D model & surgical plan: A 3D CAD model of the pelvis which is typically obtained preoperatively using a CT scan. Using this model, a surgical plan representing the pose of the implant in the acetabulum is specified for the robot arm as input.
5. Robot Arm Description: The URDF of the robot arm is used by the task prioritization controller and for collision checking.

The various inputs into the system are summarized in Figure 7.

Figure 7: The various system inputs (i) Registration Probe (ii) Pelvis EE Marker (iii) Pelvis CAD model (iv) Robot Arm CAD model

7.2.2 Perception and Sensing

The perception and sensing subsystem is responsible for receiving inputs for the various sensing modalities and the user. The perception subsystem processes these inputs, tracks the various objects in the workspace and presents feedback to some parts of the system. It also makes these inputs available to the other subsystems.

Sensing and Tracking: Primarily, the Atracsys Sprytrack 300 camera tracks predefined geometries of reflective fiducial markers. The geometries are loaded into the FrameHandler which is responsible for communicating with the linked camera libraries from within ROS, receiving tracking measurements, and continuously making them available on ROS topics for the other subsystems. The 6-DoF poses of the objects are published as standard ROS message types at 55 frames per second. The pelvis is continuously tracked by a PelvisTracker. If the change in pelvis pose is greater than the position and orientation error tolerances, the perception subsystem indicates, through a Boolean ROS message, that the pelvis pose error has been detected. In this case, the dynamic compensation is expected to be triggered. The workflow of the FrameHandler and PelvisTracker are shown in Figures 8 and 9.

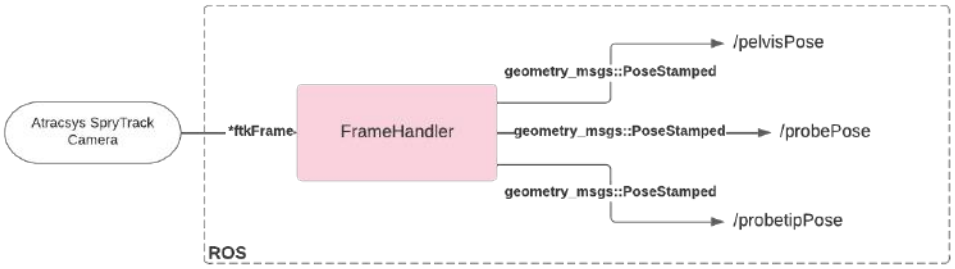

Figure 8: Frame Handler

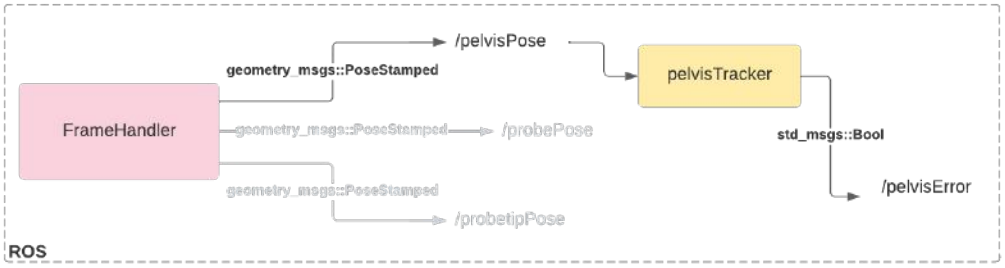

Figure 9: Pelvis Tracker

Pointcloud Collection and Registration: As an initial calibration step, the perception subsystem allows the surgeon to collect a pointcloud of key landmarks on the acetabulum. Once this point cloud is collected, the surgeon can match landmark correspondences between the preoperative scan and the collected point cloud. To simulate a preoperative scan, a pelvis sawbone model

was 3D-scanned using a Konica Minolta 3D scanner from a lab at CMU. This was then post-processed to fill in holes and eliminate other undesirable artifacts. The scan was then converted into an appropriate ASCII mesh file that could be loaded into Open3D. The yellow model in Figure 10 shows the scanned pelvis model. A sample point cloud with selected landmark points is shown on the right in 10. This enables an initial guess for the registration process. Using the Iterative Closest Point (ICP) algorithm, the transformation between the pelvis point cloud and pelvis scan is obtained. A sample registered point cloud can be seen on the right in Figure 10.

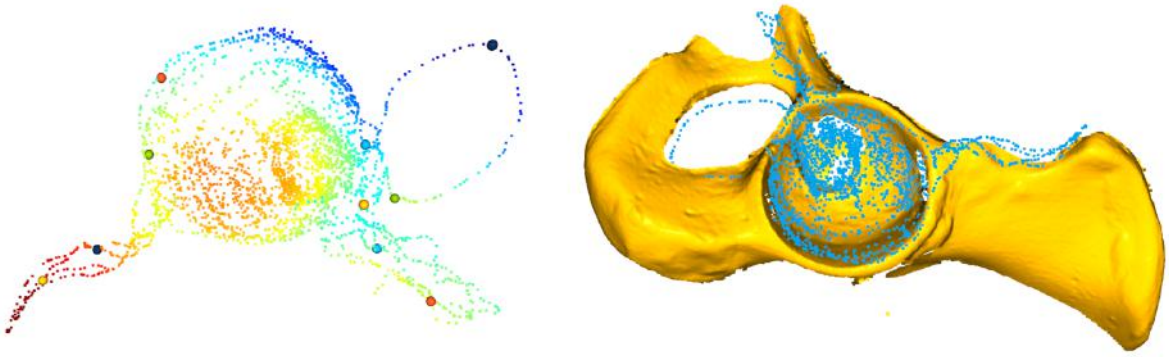

Figure 10: Collected Pointcloud Visualized in Open3D and Results of Pointcloud Registration

7.2.3 Controls

The controls subsystem is responsible for three major functions: Free Motion Mode which allows the surgeon to move the reamer end-effector close to the site of reaming before starting the surgery, Autonomous Control Mode which tracks the reaming end-pose and dynamically compensates for pelvic movement, and linearly actuate the end-effector and ream the acetabulum according to the surgical plan.

For the Kinova Gen 3 arm (7 DOF), Free Motion Mode is already implemented into the base firmware of the arm as a joint admittance mode, which can be activated by pressing a button on the last link of the arm or activated through the API calls. This made integration into our system fast and easy.

When the state machine is in Autonomous Control Mode, the control loop will take inputs from the perception system, getting the frame of the acetabulum, the arm's end-effector, and the Sprytrack Camera, allowing for the end-effector to track the acetabulum (more specifically, the reaming end-pose set by the surgical plan) while aligning the end-effector markers to the camera using a Task Prioritization Framework to execute multiple tasks at once (Figure 11). A UML diagram of the Task Prioritization Framework can be found in Figure 12. A control block diagram of the acetabular/pelvis alignment task is in Figure 13. This secondary task of aligning end-effector markers to the camera will maximize the accuracy of reaming and will enable online calibration

to allow for camera or arm movement during the surgery. In addition, the controller will also be avoiding joint limits and singularities. On a lower level, these tasks will be translated to joint velocity commands to the Kinova API. Running these tasks at high frequency will allow for dynamic compensation for any movement of the acetabulum during the operation, with the primary acetabular alignment task tracking pelvic movement.

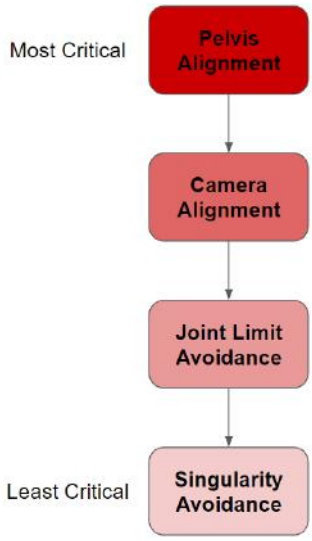

Figure 11: Tasks in Task Prioritization Framework in Order of Importance

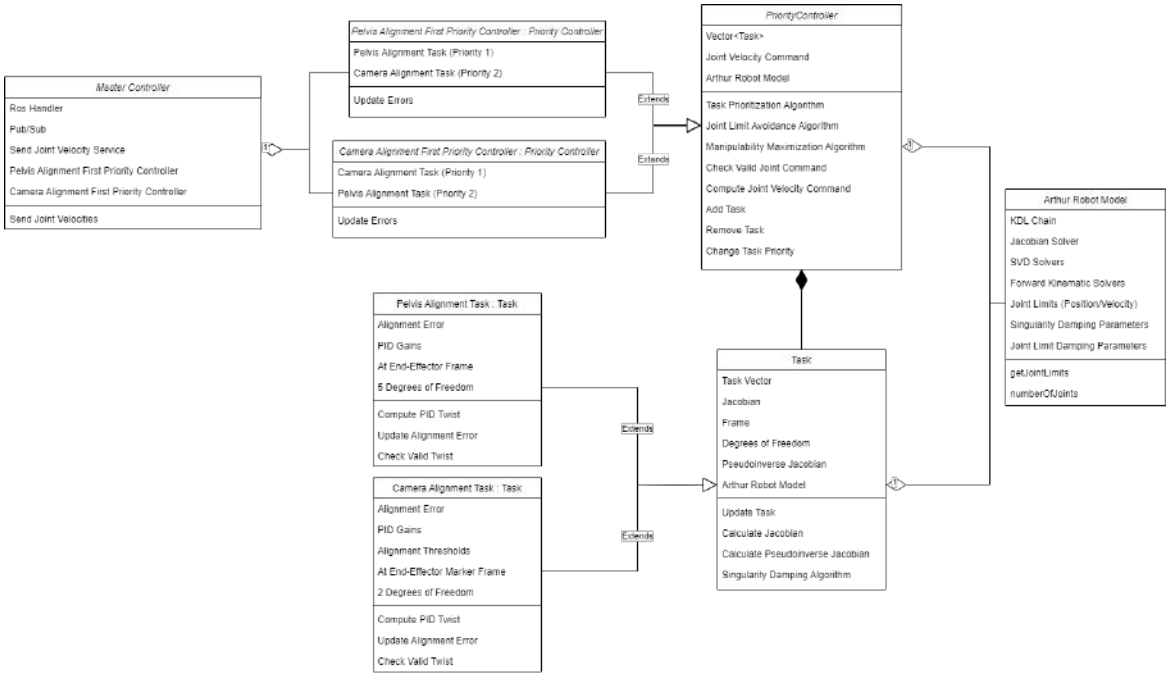

Figure 12: UML Diagram of Task Prioritization Framework

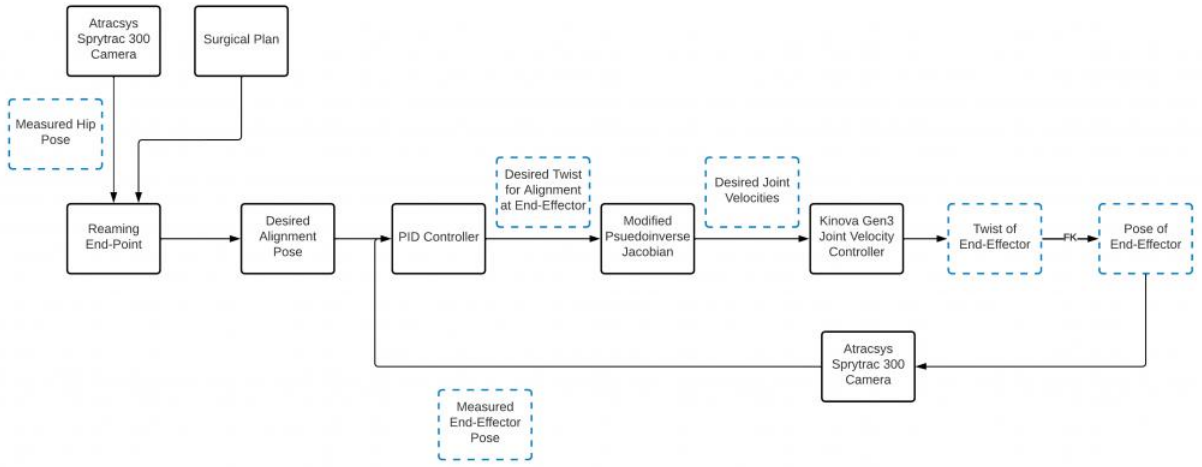

Figure 13: Acetabular/Pelvis Alignment Task Control Block Diagram

Once the task prioritization controller has aligned the robot end-effector's reamer tool frame with the desired reaming pose, and the error is within the error thresholds specified in our performance requirements, the end-effector control takes over. The general objective of this controller is to actuate the linear actuator until it reaches the reaming endpoint while maintaining a predefined amount of axial force and allowing the reamer to rotate at a predefined constant velocity. Our controller has a high-level state machine that switches between the following states:

1. **State 1 - Calibration**: This is a calibration routine wherein the linear actuator moves back until it hits the limit switch. At this point, the encoder ticks are set to zero. This initializes the position of the end-effector that will be tracked as it actuates during reaming.

2. **State 2 - Wait for command**: This is an idle state where the end-effector waits until it receives a command to start reaming from the task prioritization controls.

3. **State 3 - Move until contact**: Once a command is received for reaming, the linear actuator moves until it makes contact with a given amount of force against the acetabular surface.

4. **State 4 - Start reaming**: Once contact is made, the reamer motor turns on and maintains a constant reaming velocity until the goal state is reached or dynamic compensation becomes necessary. Once the reaming endpoint is reached, the reamer motor continues to ream for an additional thirty seconds to remove any residual bone, and also to compensate for inaccuracies caused by the end-effector and the robot arm flexing during the process.

5. **State 5 - Dynamic Compensation**: When the pelvis error exceeds the predefined position or orientation error thresholds, the end-effector retracts back to the initial state while the arm re-positions itself to the new endpoint. Once the errors are within the desired threshold, the task prioritization controller sends a signal to continue the reaming process as usual.

6. **State 6 - Finished reaming**: Final state when the reaming is completed, and the end-effector controller remains idle.

These high-level states are used to determine the motor position and velocity commands for the linear actuator and reamer motors respectively. These are executed by the low-level PID position and velocity controllers for the linear actuator and reamer motor respectively. The flowchart in Figure 14 summarizes the functionality of our end-effector controller.

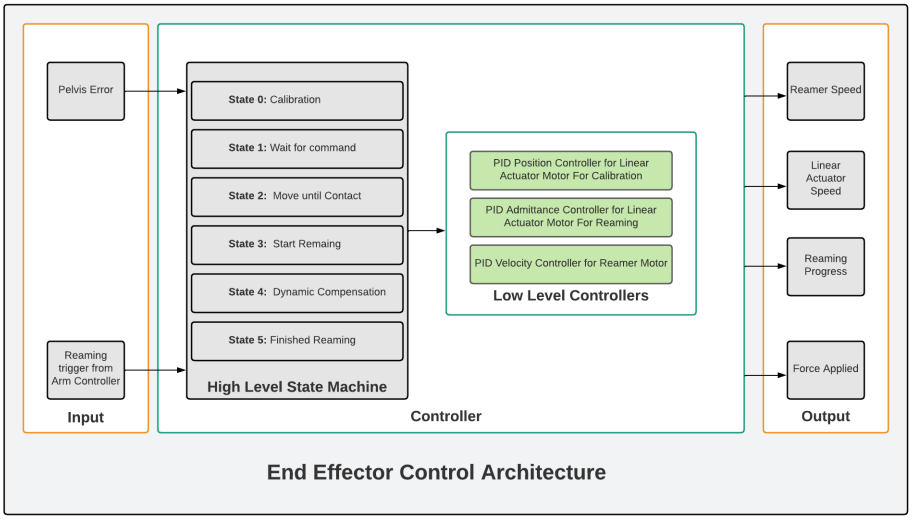

Figure 14: End-Effector Controls Architecture

7.2.4 Watchdog

The watchdog module by definition is used to detect malfunctions within the system and react accordingly based on the scale of the malfunction. In our system, the subsystems that are active in nature and need to be monitored are the controls and end-effector subsystems. Along with these, the perception subsystem is also continuously monitored because all the other subsystems are dependent on it.

The watchdog module will act as a health monitoring system and a filter for the flow of information between the subsystems. The watchdog would monitor the health of all subsystems by mainly checking the streams via which each subsystem passes information. As soon as the robot arm and the Atracsys camera are turned on, it would check for all critical functions of these subsystems such as communication between the camera and the robot, the frequency at which it is receiving information from the camera, etc. The perception subsystem sends information via ROS topics which would be monitored to detect any loss in communication. The status of each of these streams would be seen on the UI as shown in Figure 15.

If the health of the perception subsystem is good, then the watchdog will send a boolean flag to the controls subsystem indicating the health, which will allow the controller to start the end-effector alignment task. While the controller is aligning, the watchdog will keep track of the topics exposed by the controller to monitor joint limits, singularities, and the error between the desired and current orientation. If any of these parameters are abnormally wrong, the controller by itself would shut off. In case the controller doesn't shut off, the watchdog will shut the arm down. If the parameters from the controller are healthy, the watchdog will send a flag to the hardware subsystem indicating the health of the other subsystems and the progress of the alignment operation.

Figure 15: Watchdog critical parameters displayed on the UI

Once the end-effector starts the reaming process, the watchdog will track the reamer speed and the reaming progress. If the reamer is running at a speed higher than the safe threshold, then the watchdog will shut the reaming motor. Similarly, if the reamer remains on long after the reaming operation is completed, the reamer motor will be turned on.

The watchdog module also logs all system critical information in a text file that can be used for debugging or post-procedure system monitoring.

Figure 16 shows how the information is logged in a text file where the watchdog logs which marker is visible if the controller flag was set to true, if the arm is close to joint limits, etc. Not only is all this information logged in a text file but it is also displayed on the User Interface for the surgeon/user to monitor the functioning and health of the system in real-time during the procedure. Each time an error occurs, a popup window on the UI shows up alerting the surgeon about the fault in the system as shown in Figure 17.

```
Date: 22-11-17  Time: 07:41:43    Pelvis marker is not visible
Date: 22-11-17  Time: 07:41:43    End-effector marker is not visible
Date: 22-11-17  Time: 07:41:43    Controller flag couldn't be set to true because downstream processes are unhealthy
Date: 22-11-17  Time: 07:41:43    Controller error publisher dropped below 30Hz
Date: 22-11-17  Time: 07:41:43    Controller singularity publisher dropped below 30Hz
Date: 22-11-17  Time: 07:41:43    Controller joint limits publisher dropped below 30Hz
Date: 22-11-17  Time: 07:41:43    Controls translation error is high
Date: 22-11-17  Time: 07:41:43    Controls orientation error is high
Date: 22-11-17  Time: 07:41:43    Arthur at singularity
Date: 22-11-17  Time: 07:41:43    Hardware flag set to true
Date: 22-11-17  Time: 07:41:43    Pelvis is visible
Date: 22-11-17  Time: 07:41:43    End-effector is visible
Date: 22-11-17  Time: 07:43:27    Controls translation error is within limits
Date: 22-11-17  Time: 07:43:27    Controls orientation error is within limits
Date: 22-11-17  Time: 07:43:27    Controller flag set to true
Date: 22-11-17  Time: 07:43:27    Arthur out of singularity
Date: 22-11-17  Time: 07:43:27    Hardware flag set to false
Date: 22-11-17  Time: 07:43:27    Controller flag couldn't be set to true because downstream processes are unhealthy
Date: 22-11-17  Time: 07:43:27    Controls translation error is high
Date: 22-11-17  Time: 07:43:27    Hardware flag set to true
Date: 22-11-17  Time: 07:43:33    Pelvis marker is not visible
Date: 22-11-17  Time: 07:43:33    Pelvis is visible
Date: 22-11-17  Time: 07:43:39    Controller flag set to true
Date: 22-11-17  Time: 07:43:39    Controls translation error is within limits
Date: 22-11-17  Time: 07:43:39    Hardware flag set to false
Date: 22-11-17  Time: 07:47:47    Pelvis marker is not visible
Date: 22-11-17  Time: 07:47:47    Controller flag couldn't be set to true because downstream processes are unhealthy
Date: 22-11-17  Time: 07:47:47    Hardware flag set to true
Date: 22-11-17  Time: 07:47:47    Pelvis is visible
Date: 22-11-17  Time: 07:47:47    Controller flag set to true
Date: 22-11-17  Time: 07:47:47    Hardware flag set to false
Date: 22-11-17  Time: 07:47:48    Pelvis marker is not visible
Date: 22-11-17  Time: 07:47:48    Controller flag couldn't be set to true because downstream processes are unhealthy
Date: 22-11-17  Time: 07:47:48    Hardware flag set to true
Date: 22-11-17  Time: 07:47:50    Pelvis is visible
Date: 22-11-17  Time: 07:47:50    Controller flag set to true
Date: 22-11-17  Time: 07:47:50    Hardware flag set to false
Date: 22-11-17  Time: 07:57:48    Pelvis marker is not visible
Date: 22-11-17  Time: 07:57:48    End-effector marker is not visible
Date: 22-11-17  Time: 07:57:48    Controller flag couldn't be set to true because downstream processes are unhealthy
Date: 22-11-17  Time: 07:57:48    Hardware flag set to true
```

Figure 16: Watchdog logging system faults during the procedure

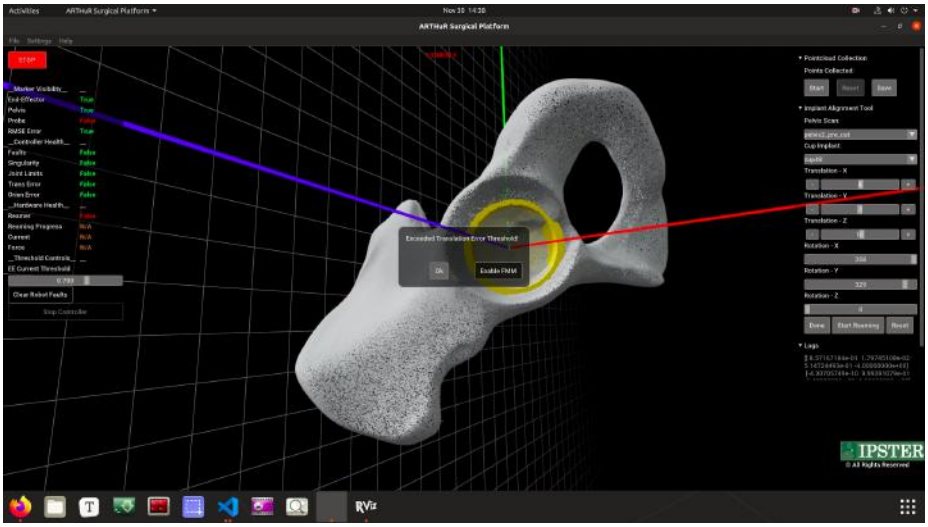

Figure 17: Fault alert popup on the UI

The system is able to resume operation only if the fault is cleared. If any of the parameters are unhealthy during the procedure, the controller flag is set to false which stops the controller from taking further actions and the emergency stop is initiated. Similarly, the end-effector is also stopped in case of emergencies.

7.2.5 Hardware

Set-up of the hardware subsystem began with collecting and installing parts of the system from our sponsors and other sources within the Robotics Institute. In total, our sponsors provided us with an Atracsys Sprytrack camera, fiducial markers and their associated mounts, a reamer handle, reamer heads, and the Kinova Gen-3 robotic arm. We also received a Vention table from Professor Kroemer and a force torque sensor from Professor Held. With all these components collected, the

majority of our work for the hardware subsystem came in designing, prototyping, and manufacturing the end-effector, and designing an electrical system to control it.

The reaming end-effector design went through several evolutions throughout the project. In the spring semester, we designed a simple end-effector which was simply a motor axially aligned with the last joint of the arm which rotated a reamer shaft. This original design, which can be seen in Figure 18 worked okay as it could actually cut our Sawbone pelvises as necessary, but it introduced many problems into our system. Due to its design, it required the robot arm itself to move in order to provide a reaming force into the acetabulum, which caused lots of vibrations in the system, while simultaneously decreasing the degrees of freedom of the system due to the axial alignment of the end-effector. Moving into the fall semester we knew we wanted to make a change.

Figure 18: Spring Semester End-Effector

This led us to redesign the end-effector entirely. To solve these problems, we decided to move towards an angled linearly actuated design which allowed us to solve these problems. With an angled design we are no longer losing degrees of freedom and forcing the arm into awkward positions, and with a linearly actuated design, we allow the arm to only position the end-effector and perform dynamic compensation, instead of relying on it to also apply force to the acetabulum for reaming. Our design utilizes a linear rail actuated via a ball screw, DC planetary gear motors with encoders, limit switches, and a reamer shaft which we received from our sponsors and cut down to size such that it could interface with one of our motors via a coupling. These components can be seen in Figure 19 which summarizes all the different components that comprise our hardware system.

In the fall semester, we iterated on this design several times, going through many 3D printing prototype phases until we reached a design we were comfortable with. At this point, because of our desire to reduce vibrations and develop a more professional final product, we decided to manufacture some of our structural components out of aluminum. Due to part complexity, low costs, and fast shipping, we decided not to manufacture these components ourselves in the RI Machine Shop, but instead to utilize Xometry which required us to generate engineering drawings of all

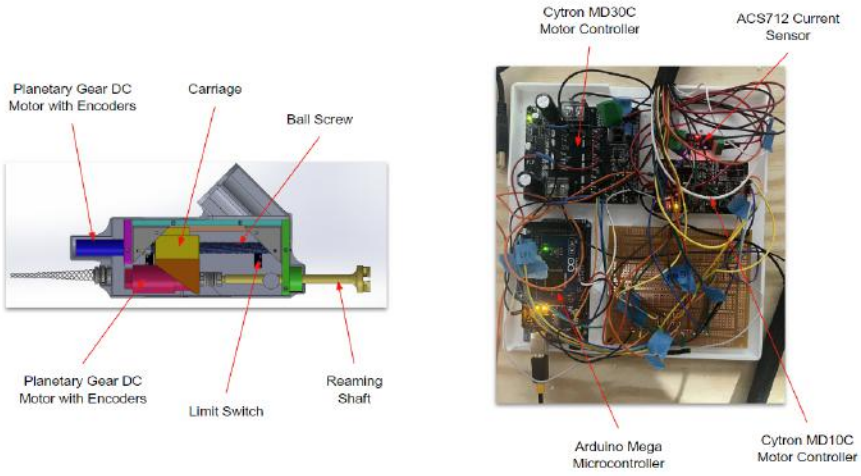

Figure 19: Hardware Components

of our components to specify holes and tolerances. Thankfully, these parts came in quickly and turned out perfectly for us, requiring no changes once they arrived and fitting together exactly as desired. The resulting end-effector can be seen in Figure 20.

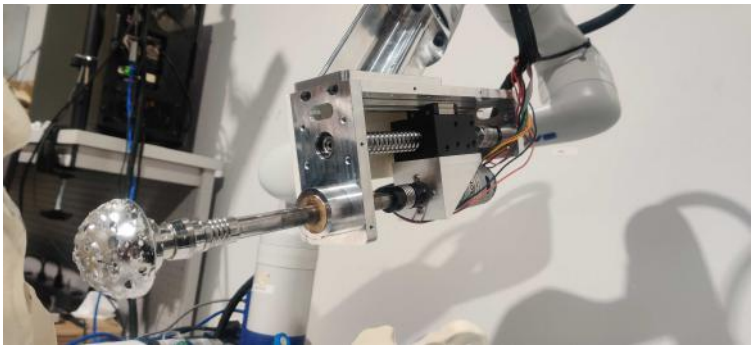

Figure 20: Assembled End-Effector without Cover

With this assembled we could now focus on designing a cover for the end-effector, allowing for the mechanical system to be enclosed for faux sanitization purposes. The cover was designed to be 3D printed as eight different parts, two of which attach to the sides of the aluminum components, and six of which attach to those side parts via embedded magnets, allowing for quick and easy disassembly. These covers also included holes for adapting our fiducial markers, which could be fixed onto the side of our end-effector by 3D-printed external threads. A picture of the final end-effector can be seen in Figure 21.

Then came the time to design the electrical system to interface with the hardware and control it as desired by our controls subsystem. Spring semester we utilized a custom PCB and a Cytron motor controller for our electrical subsystem, but this semester we decided to revise our subsystem. We decided it would be a lot easier to utilize an Arduino Mega microcontroller as the heart of our electrical system and use that to control the motors in the end-effector. All low-level force and

Figure 21: Final End-Effector with Cover

velocity control of the reamer would be handled by the Arduino Mega which would communicate with our main system, receiving commands to begin reaming and stop reaming, and reporting back important values. The Arduino interfaces with the motors via Cytron motor controllers (one MD10C and one MD30C due to issues with current spikes) and receive feedback from the encoders built into the DC motors. The Arduino also receives sensor information from limit switches in the end-effector (allowing for calibration and sensing of over-actuation) and current sensors which indirectly measure the axial force being applied by the end-effector. Figure 22 displays a block diagram of our electrical subsystem, and Figure 19 shows how the components are laid out in our 3D-printed electrical box.

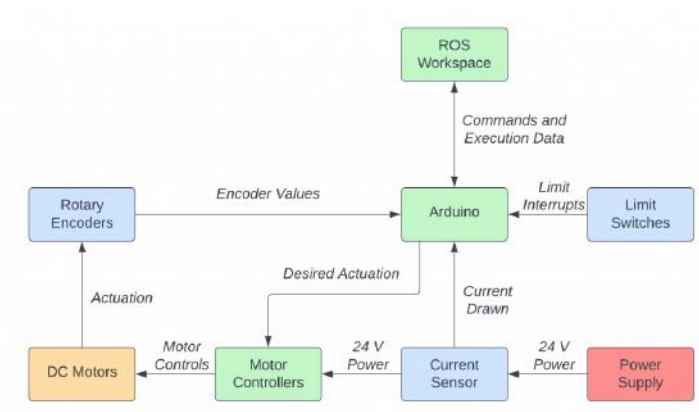

Figure 22: Electrical System Diagram

7.2.6 User Interface

The user interface, shown in Figure 23, is a desktop application containing a set of tools that enable the surgeon to communicate with the system. It is developed using Open3D's GUI library. Open3D allows for easy manipulation of multiple pointclouds and provides a set of tools that help develop a simple, yet functional UI frontend. The first step in development was creating wireframes. These wireframes depicted all the required elements and functions of the UI. Next,

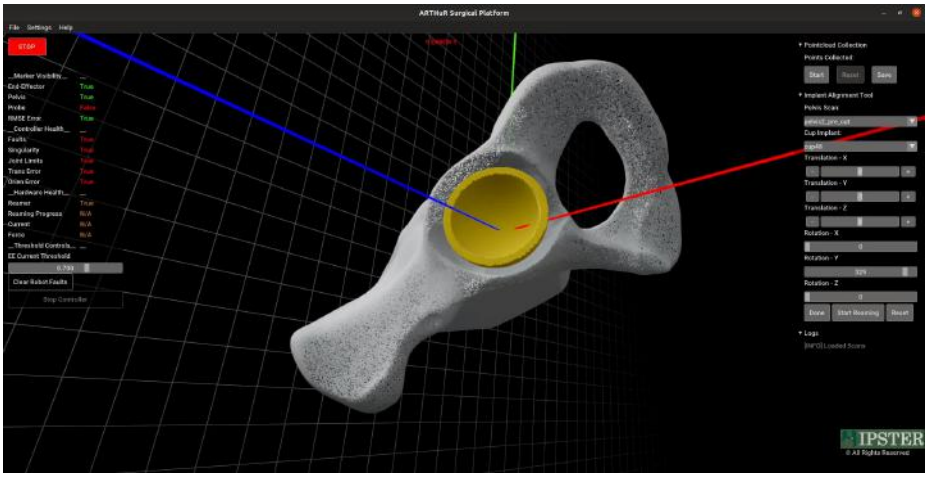

Figure 23: User Interface

a simple prototype of the UI was developed that is capable of manipulating 2 3D objects. Since this was the most crucial functionality, it was important to ensure that multiple 3D geometries can be loaded onto a scene and transformed without significant latency. Once the prototype was complete, we developed the final UI frontend. Several front-end changes were made to ensure the best UI/UX. Next, the UI was integrated with ROS in order to communicate with other subsystems. This involved a careful understanding of event callbacks and threading. Below are the primary functionalities of the UI.

Landmark Point Collection and Matching The user interface enables the surgeon to start, stop and pause the collection of landmark points on the pelvis. At any moment, during the collection of points, if the surgeon requires resetting the process, the UI provides a reset option to do so. The number of collected points is displayed in real-time on the screen for guidance.
Once the landmark points are collected, the landmark matching process is triggered. This allows the surgeon to match landmark points on the preoperative pelvis scan and the collected landmark points. Once the registration process is complete, the results can be visualized and the surgeon may decide to proceed with the surgical plan or redo the landmark collection/selection process.

Implant Alignment Tool The surgeon is able to plan the surgery using the implant alignment tool. First, the UI loads the selected preoperative pelvis scan and corresponding cup implant in 3D. This enables the surgeon to visualize the relative position of the cup implant with respect to the pelvis. Using a set of sliders and buttons, the surgeon is able to apply small transformations to the cup implant in order to align it with the pelvis' acetabulum. During alignment, the UI allows viewing the objects from several angles, zooming, panning of the views, and so on. Once the surgeon is satisfied with the alignment, they can communicate the same with the other subsystems. The UI broadcasts (via ROS) the relative pose of the cup with respect to the pelvis as a 6DoF pose. This is shown in Figure 24.

System Health Monitoring and Metrics As a time and health critical system, it is essential that the surgeon is able to constantly monitor the system and learn of any abnormalities. These

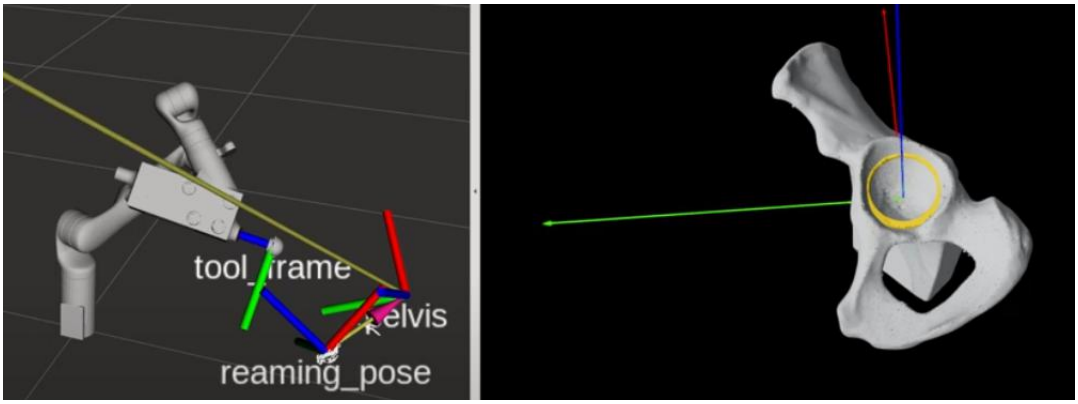

Figure 24: [Left] Reaming pose broadcasted by ROS [Right] User interface showing the surgical plan created by aligning the cup implant with the pelvis' acetabulum.

may include situations such as the tracking markers being blocked, control subsystem errors, and so on. In these cases, the surgeon can easily toggle the system between free-motion mode and autonomous mode. Furthermore, in case of an emergency, the surgeon can estop the system.

7.3 Modeling, Analysis, and Testing

7.3.1 Hardware Analysis

As we went through designing our hardware for this project, we set up a lot of requirements that our end-effector needed to meet in order to properly work with the rest of our system and safely perform reaming operations. Thus, throughout the design process, we had several tests that verified that our design was fitting our requirements. Our first tests simply determined that our initial 3D printed prototypes allowed for greater than 50 millimeters of actuation via the ballscrew, that the end-effector remains solidly attached to the robot arm at any orientation, and that the vibrations are minimal, all of which were verified before moving on. Next, we tested that with our electrical system set up with our 3D printed end-effector prototype, our motors can turn on and actuate as desired, our velocity can be reported via a ROS topic, our axial force can be reported via a ROS topic, and that we report no errors during tests. All of these simple tests were successes after much integration. Our final test during the design and integration part of the project verified that our hardware was properly integrated with the rest of the system. This meant that our motors were able to be turned on and off via ROS, our velocity and force were being properly communicated to ROS and utilized by it, our motors stop within 500 ms of receiving a stop command, and that dynamic compensation does not affect our ability to ream. All these tests were determined to be successful by the end of integration, although importantly we decided to hook our power supply through an e-stop so that we could stop the end-effector faster as necessary in an emergency by completely cutting power from it.

With all these tests passed, it came time to analyze the performance of the system against what we desired for it. The first test was that our end-effector doesn't exceed 100 N during a reaming operation as exceeding that amount of force risks damaging a patient's pelvis. The experiment we ran to verify this was admittedly a little crude and involved positioning our end-effector to point directly into a scale at which point we turned on our reamer and allowed it to press against the scale

for a few minutes. This experimental setup can be seen in Figure 25. Through this experimentation, we determined that the max force applied was around 62.3 N and the mean force applied was 37.6 N. Thus, our end-effector actuates safely within the force thresholds we have set.

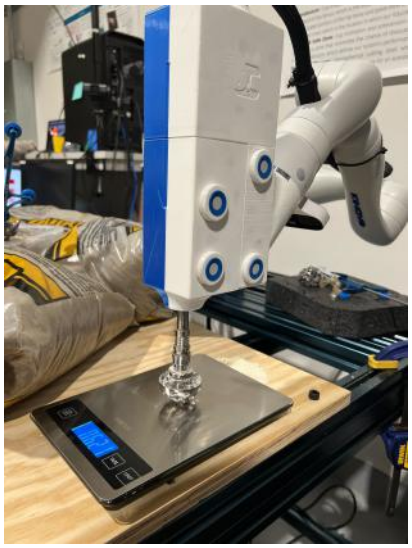

Figure 25: Testing End-Effector Force

Another requirement of our end-effector was that it remains under 4 kilograms, as exceeding that weight would lead to the collaborative robot arm having difficulty moving as we need it to. This was a concerning requirement and led us to carefully design our aluminum structural components to be as thin as possible with some extra material taken out to save weight. Thankfully our efforts were successful and the final weight of the end-effector was around 2.8 kilograms. Another requirement of the end-effector was that it reduces vibrations as much as possible within the end-effector. This was proven to be the case during experimentation as vibrations were minimized within the end-effector itself. However, this was due to the arm itself vibrating and adjusting during the reaming operation. Due to the collaborative nature of the arm, it isn't very rigid and thus the arm itself held our system back from being more rigid. The final requirement for our end-effector was that our linear positioning is accurate to 0.5 mm. Ideally, due to the encoders in our motors and the pitch of our ballscrew, we should be accurate to around 0.02 mm or one encoder tick. However, because of the non-linearity of controlling the motor through PWM, we are only accurate to five encoder ticks or 0.1 mm, which is still a success. Our hardware passed every test and requirement set for it this semester which provided us with a robust and accurate final result.

7.3.2 Controls Analysis

The Task Prioritization Framework is fundamentally about projecting lower-priority tasks into the null space of higher-priority tasks to prevent those lower-priority tasks from interfering with the performance of the higher-priority tasks. For instance, we project the camera to end-effector marker alignment task into the null space of pelvis/acetabular alignment task. In order to implement the algorithm to support several levels of prioritization, we had to generalize the algorithm

found in literature [4]. Generalizing the algorithm, we got the following Task Prioritization Algorithm (Equation 1).

$$
\begin{aligned}
&\dot{q} = \sum_{i=1}^{n} \dot{q}_i, \text{ where} \\
&\dot{q}_i = (J_i(\prod_{j=1}^{i-1} N_j))^{\#}(\dot{x}_i - J_i(\sum_{j=1}^{i-1} \dot{q}_j)), \\
&N_j = I - (J_{j-1}N_{j-1})^{\#}(J_{j-1}N_{j-1}) \\
&N_1 = I, \\
&\text{and n is the number of tasks, where highest priority is i = 1.}
\end{aligned}
\tag{1}
$$

Using this framework, four tasks were implemented with importance in the following order, highest to lowest: acetabular/pelvis alignment, camera-to-end-effector marker alignment, joint limit avoidance, and singularity avoidance. We also implemented collision detection as a safety check.

We performed several tests to evaluate the performance of our controller. One of the first tests we did was to ensure that the steady-state error of the end-effector position for tracking a frame in space was zero. To do this, we attached a static frame with respect to the pelvis markers and had the arm track this static frame as we moved the pelvis. Using the camera, we tracked the end-effector's error with respect to the pelvis. Figure 34 (in the Appendix) shows a plot of the error of tracking the pelvis as it moves. The spikes in error are when the pelvis is moved, and after a few seconds, the steady-state error goes back to zero. A similar test was performed for the camera-to-end-effector marker alignment with an initial goal of staying under 10 degrees of alignment error between the end-effector marker and camera. However, because this task had to sacrifice a degree of freedom to the higher priority acetabular/pelvis alignment task, it is unable to get below that error in some configurations. Figure 35 (in the Appendix) shows the steady-state error of aligning the orientation of the end-effector markers to the camera over time. The error is minimized at all configurations given the constraint to align to the pelvis first. Figure 26 shows the end-effector adjusting to the camera movement while keeping alignment with the acetabulum.

While the end-effector aligns the markers to the camera, online calibration is happening in parallel, using the end-effector's marker and joint encoder angles to estimate the position of the base with respect to the camera in real-time. Using the camera and registration probe measurements, the estimated error from online calibration is less than 1 mm. As previously stated, joint limit avoidance and singularity avoidance were also implemented as repulsive potential fields and were tested. Figure 27 shows singularity avoidance and joint limit avoidance tasks being tested for effectiveness. As seen in the figure, the arm moves away from singularities, therefore maximizing stiffness equally in all directions at the end-effector. The arm also avoids joint limits by using other joints to perform the inverse kinematics, not using joint 2 anymore as shown in the figure to avoid hitting joint 2 limits.

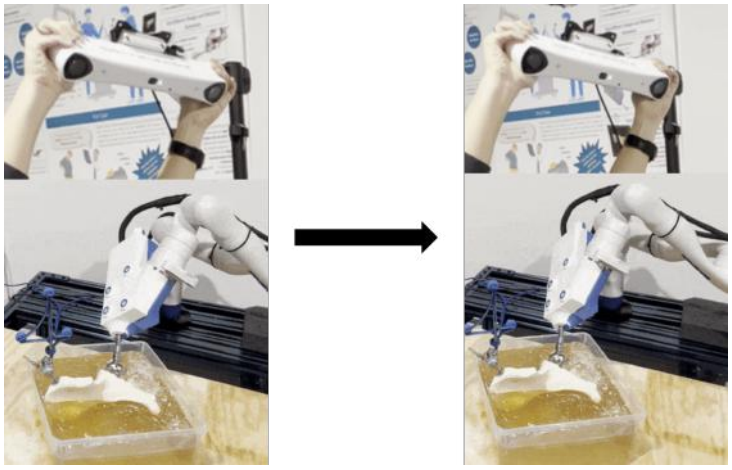

Figure 26: Camera to End-Effector Marker Alignment Task

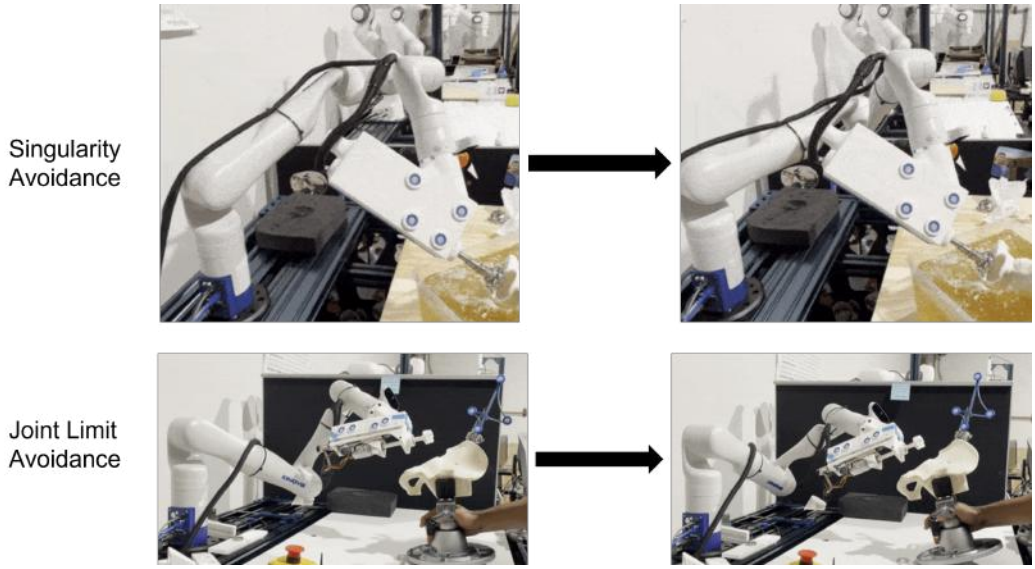

Figure 27: Singularity Avoidance and Joint Limit Avoidance Tasks

7.3.3 Watchdog Analysis

The watchdog module is responsible for maintaining safety for the users and surgeons. In case of any faults occurring in any of the subsystems, the watchdog stops the controller, stops the end-effector, and pushes the arm to an emergency state. The watchdog module needs to run at a rate higher than all the other subsystems. The camera streams data at a rate of 55 Hz, which is the fastest subsystem. Hence, the watchdog runs at a rate of 100 Hz. It could also run at a higher rate, for example at 1000 Hz, if necessary. However, in the current setup, a loop rate of 1000 Hz bottlenecks the Arduino used in the end-effector.
We performed extensive testing to realize all the cases in which a fault might occur and corrective action needs to be taken by the surgeon. We tested the system inputs such as the reflective markers on the end-effector, markers on the pelvis, and the registration probe. The registration probe should not be visible at all times. However, if any of the other markers becomes invisible even for a split

second, the controllers are paused and an emergency stop is initiated. We also conducted extensive testing to threshold the singularity limits such that we don't stop the controller prematurely. Listed below are all the cases we have tested under which the watchdog stops the controller and initiates an emergency stop:

- End effector not visible — Pelvis not visible
- Registration RMSE error too high
- Arm at singularity — Arm at joint limits
- Translation error too high — Orientation error too high
- Excess current drawn by reamer motor — Excess force applied by end-effector — Reaming progress over 100%

Based on the learnings from all the testing, we introduced smaller features such as providing the user 10 seconds to correct a fault and waiting for the user to press 'Clear Faults' on the user interface to continue with the procedure. These features not only helped us conduct experiments in a smooth fashion but also would help any users of the system have control over the system while staying safe around the robot.

7.4 FVD Performance Evaluation

To validate our final results we decided to directly compare what we desired to ream and what we actually reamed, which would provide us a better understanding of the accuracy of our system. We did this by first scanning our pelvis prior to reaming using a Creaform HandyScan which provided us with an STL of the pelvis prior to reaming. We then went through the reaming procedure and made sure to save the specified orientation of the cup with respect to the pelvis that was applied during reaming. Using Blender, the desired cup position was subtracted from the pelvis, providing us with the desired pelvis to compare to. We then scanned the reamed pelvis using the HandyScan, so that now we had our desired and our actual pelvis. Using CloudCompare we then compared the acetabulums of these two pelvises to obtain the results as summarized in table 3 and seen in Figure 28.

Our goal was for every operation we performed to be within 2 mm of the specified position and orientation at worst, which we managed to hit in 2 out of the 5 experiments we ran. For the other three experiments, as can be seen, we were at worst 3.4 mm off, and our averages for all trials were less than 1 mm. Analyzing the figures in Figure 28 reveals that often we were reaming eccentrically from where we wanted to be reaming, as it can be seen that there are patches where we reamed too deeply on the opposite side of patches where we did not ream deep enough, which a patch in the center which was reamed perfectly. This feeds the idea that we were often reaming to an accurate depth, however, our accuracy may have been off as we moved into the acetabulum. This could have been from a variety of small factors, but the two most likely are bad ICP registrations and elasticity in the robot arm. With a poor ICP registration that we did not detect it is possible that where the computer thinks the pelvis is in space and where it actually is is slightly offset, leading to this error. We attempted to combat this by taking points for our point cloud of the

Table 3: Validation Results

Test Number	Maximum Signed Error	Average Error
1	+2.0 mm	0.29 mm
2	- 3.0 mm	0.43 mm
3	+3.4 mm	0.95 mm
4	-1.3 mm	0.42 mm
5	+3.3 mm	0.52 mm

Iliac crest, which did help, but our point cloud was definitely still offset from the actual pelvis at times. The larger contributing factor to this inaccuracy is likely the elasticity in our collaborative robot arm. What we believe happened from analyzing the actual reaming operations is that when the end-effector was actuated to attempt to perform slightly eccentric reaming (taking more material off of one side of the acetabulum than the other) instead of doing so and taking material off the side, the forces in the end-effector led the arm itself to bend and allow the end-effector to slip into the and concentrically ream from there. This explanation makes sense with the consistency with which the same eccentric reaming problem can be seen in our validation results, as one side ends up getting reamed more and one side gets reamed less as a result of this slipping and bending.

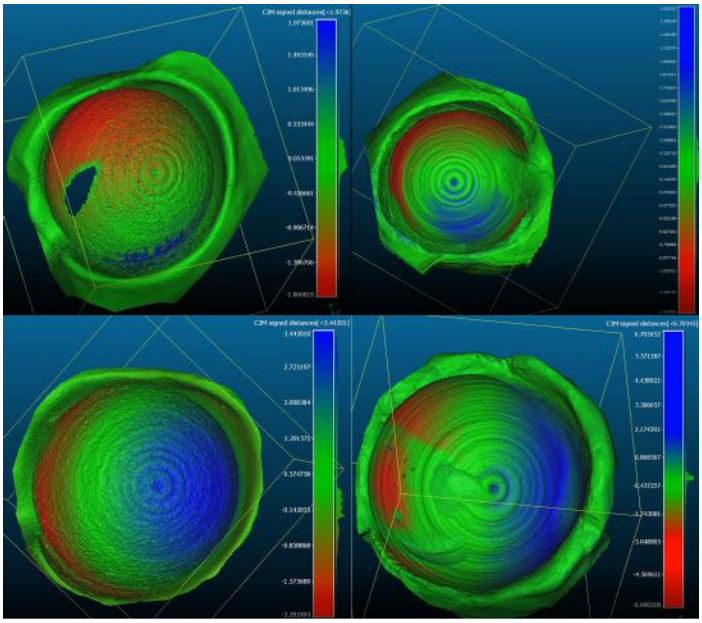

Figure 28: Validation Results Visualizations

7.5 System Strengths and Weaknesses

Our progress through the semester led to a successful demonstration of each individual subsystem and a demonstration of the integrated system as a whole. As with every technical challenge, our system also has some strengths and weaknesses as listed below.

7.5.1 Perception & Sensing

- **Strengths:** The sub-system uses start-of-the-art methods for registration and is robust to outliers in the point cloud collection. It also has very low latency and high accuracy.
- **Weakness:** Sometimes suffers from bad point cloud collections and needs a greater number of points for good registration.
- **Future work:** Reduce the number of points for registration and increase the repeatability of point cloud collection.

7.5.2 Controls

- **Strengths:** The sub-systems are highly accurate and repeatable in their performance, and the controller can execute multiple critical tasks at once in a prioritized manner.
- **Weakness:** The Kinova API is bottlenecked at 40 Hz, so the controller is not very fast at tracking the pelvis.
- **Future work:** Use Kinova's lower-level API to increase the controller rate to 1 kHz.

7.5.3 Hardware

- **Strengths:** The system is designed with modularity and ease of use in mind and the final product has proven itself to be robust and accurate.
- **Weaknesses:** The electrical system is not integrated onto a single PCB yet, and the collaborative robot arm holds back our potential accuracy as it flexes and vibrates during operations.
- **Future work:** Develop a control PCB for the system, and integrate a new robot arm into the project which is more robust and resistant to flexing as the Kinova did.

7.5.4 User Interface

- **Strengths:** The user interface is robust and currently works reliably.
- **Weakness:** Several medical applications are built on specific software that allows for cross-sectional views of 3D scans. Currently, the UI only shows the whole 3D object in space.
- **Future work:** We must integrate the User Interface with UI/UX design principles to improve the overall user experience. Further, the implant alignment can be done more intuitively by showing the movement of the joint in varying implant positions.

8 Project Management

8.1 Schedule

Our schedule for Fall 2022 can be seen in Figure 28. We divided the tasks for this semester based on each subsystem, which included improvement and changes to existing subsystems such

as controls and end-effector. Additionally, we also planned a schedule for completely new subsystems such as the watchdog and user interface which had a slightly longer development cycle. This semester, we moved away from fixed biweekly sprints as we had in the spring semester and decided to estimate the time required to complete a task and have that as a milestone instead. Having a milestone for each task helped us simplify the scheduling process. In the process of estimation for completing a task, we also made sure to include a buffer period for unforeseen events.

We were successful in creating a schedule that worked well for our team. Even though we did not follow the agile processes of sprints, the creation of milestones for each subsystem helped us follow our plans. A simpler schedule helps in not getting overwhelmed by the many deadlines. As we did not have a dedicated project manager to track the schedule, each subsystem owner along with the co-owner was responsible for tracking the milestones, which worked well for us. We visited the schedule in our weekly standup meetings. The difficulty in our scheduling process was the estimation of time to complete a task. We often found ourselves needing to use the buffer time we had scheduled for a subsystem or sometimes even exceeding the buffer time. This is mainly because of the lack of information about the complexity of the subsystem and the time spent in debugging any troubles.

8.2 Budget

Our total project cost is $5,615. We exceeded our budget by $615. Our primary expenditure was manufacturing our end-effector in aluminum, which we outsourced to Xometry. This also included spare parts in case of failure or breakage of any parts. As is the case in any custom design, we also spent some money identifying the right motors for our use case. The primary reason for our exceeding our budget is having spare parts for all critical parts of the system such as motors, power supply, motor drivers, Arduino, etc. A comprehensive list of parts ordered and use during the course of the project can be seen in Table 5. Table 4 shows all the parts that we borrowed with their estimated cost.

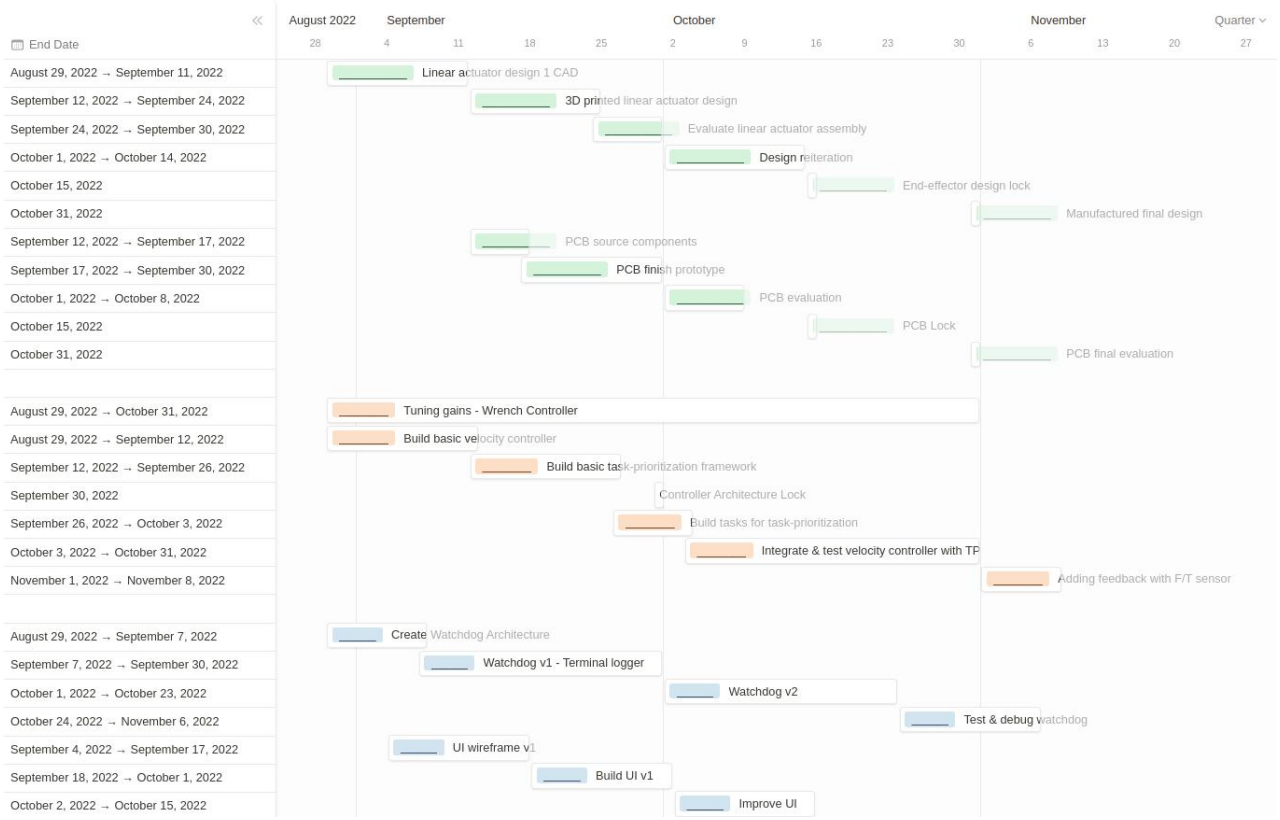

Figure 28: Fall Schedule and Milestones

Table 4: Borrowed Parts List

Part Name	Description	Date Received	Supplier	Location Stored	Estimated Cost
Reflective Markers and Marker Probes	Markers that Atracsys Camera System can track to enable localization of pelvis and arm	November 10th, 2021	Sponsor	NSH B512	$500
Atracsys 300 Camera	Camera for perception system	November 10th, 2021	Sponsor	NSH B512	$25,000
Vention Mounting Table	Mounting table to mount hardware to	February 18th, 2022	Oliver Kroemer	NSH B512	$2,000
Kinova Gen 3 Arm	Manipulator arm which is the main hardware for our system	February 28th, 2022	Sponsor	NSH B512	$40,000
MRSD Desktop 2	Desktop System to run our systems on	February 7th, 2022	MRSD Inventory	NSH B512	$2,000
Camera Mount Adapter	Mounting hardware to mount camera to suitable position. We fabricated this ourselves.	January 21st, 2022	RI Machine Shop	NSH B512	$50
Force/Torque Sensor/Accessories	Force/torque sensor to allow us to see how much force is applied to the bone	March 11th, 2022	David Held's Lab	NSH B512	$5,000
PCB and Power Distribution Hardware (Arduino Nano, 12V Power Supply, Pololu Motor Driver)	Power distribution system to drive motor at end effector, enabling reaming	March 28th, 2022	MRSD Project Course	NSH B512	$100
Reamer Heads and Handles	Reamer heads that can be attached to the motor to allow reaming of bones	March 29th, 2022	Sponsor	NSH B512	$1,000
Cytron MD10C Motor Controller	Motor controller to replace the Pololu 1457s	April 11th, 2022	MRSD Inventory	NSH B512	$100

Table 5: Parts List and Total Expenditure

Description	Quantity	Brand	Model	Cost
Pelvis Bone Models	16	Sawbone		$316
VIVO Single Monitor Desk Mount	1	Vivo		$40
612 RPM HD Premium Planetary Gear Motor w/Encoder	3	Servo CIty		$180
Pololu High Power Motor Driver	1	Polulu		$84.95
Arduino Nano	2	Arduino	Nano	$33
Power Supply Adapter DC 12V 20A	2	GESD		$50.00
Power Supply Adapter DC 12V 30A	1	GESD		$26.41
Pololu High Power Motor Drvier	2	Polulu		$181.50
Ethernet Splitter	1	TP-Link		$17.99
Power Strip	1	GE Home Electrical		$14.56
Surgeon gown	1	BH SUPPLIES		$27.99
Presentation laser pointer	1	Laser Pointer		$29.99
Conference mic	1	Tonor		$32.99
Angle Vise	1	McMaster-Carr		$431
100 mm linear actuator using ballscrew and stepper	1	RATTMMOTOR		$97.90
116 RPM Planetary Gear Motor	2	Servo City		$100.00
Miniature Tension Load Cell (50 kg option)	2	ATO		$344.98
Load Cell Junction Box (2 Inlets to 1 Outlet option)	1	ATO		$66.67
Digital Load Cell Amplifier (0-5V option)	1	ATO		$109.85
Limit Switch	1	MXRS		$6.99
LocknLock Easy Essentials Airtight Rectangular Tall Food Storage Container 152.16-oz / 18.8 Cup, Clear	2	LocknLock		$29.98
Knox Unflavored Gelatin Mix (Bulk) Container, 16 OZ (Pack of 4)	2	Knox		$124.10
2x30A Current Sensors	1	NewZoll		$10.29
Xbox controller for robot	1	PowerA		$25
Vibration dampening bearing	1	McMaster-Carr		$148.58
Xomerty End-effector Parts	1	Xometry		$1,342.22
Neodymium magnets	120	McMaster-Carr		$124.40
Team Swag	7	Custom Ink		$253
520 RPM Premium Planetary Gear Motor w/Encoder	1	Servo City		$50
Arduino Mega	1	Arduino	Mega	$48.40
Premium Planetary 313 rpm gear motor	1	Servo City		$59.99
Cytron MD10C	1	Cytron	MD10C	$17.50
165 RPM HD Premium Planetary Gear Motor w/Encoder	1	Servo City		$60
Cables, Cords, & Sleeves	1	N/A		106.96
Bolts & Nuts	1	McMaster-Carr		$128.15
Electrical connectors & Pins	1	N/A		$196.15
Motor Coupling	7	McMaster-Carr		$482.42
Miscellaneous	1	N/A		$215.01
Total Project Cost				**$5,615**

8.3 Risk Management

8.3.1 Overall Process:

Since the beginning, we have been diligent and conscious of the various risks in our project. Our process involved identifying and periodically evaluating our project risks as a team. These risks were broadly classified into technical, programmatic, and schedule-related risks. Each one was given a score from 1-5 on its likelihood of occurrence (1 being least likely) and adversity of its consequences (1 being least adverse). Each risk had a set of mitigation strategies associated with it to reduce the chances of the risk affecting the outcomes of our project. We documented and maintained our risks on a common spreadsheet to facilitate better visibility and collaboration. To better manage and keep track of risks at all times, one team member was also made responsible for ensuring that the mitigation strategies discussed are being followed. Throughout the spring semester of 2022, we were actively tracking a total of 11 risks, of which two were critical risks (i.e., those with a likelihood and/or consequence greater than 4). In the fall semester of 2022, we were actively tracking a total of 17 risks, of which 5 were critical risks. Figure 29 below

summarizes our critical risks and the actions taken to mitigate them successfully.

8.3.2 Successes:

In retrospect, we believe we were successful in our risk management and mitigation strategy. We had four critical risks that could have severely derailed the project's outcomes. In the spring semester, our critical risk involving the robot arm getting delayed was realized. However, we mitigated this risk by focusing heavily on development using our simulation environment. The second major risk was ensuring that the robot arm and camera provided by our sponsors do not get damaged. Having a good simulation was necessary since we were developing a custom inverse-kinematics library and using it to override the manufacturer's high-level APIs that already have a layer of safety built-in. We also maintained documentation of best practices and ensured that all team members were aware of the standard operating procedures when using any of the hardware provided to us. Our constant obsession with safety also led to the development of a watchdog module, which is a layer of safety that ensures each system parameter is within the expected bounds. d Thirdly, we factored in the various schedule delays that could occur, especially when dealing with third-party vendors with our custom end-effector. We successfully tested multiple iterations of our end-effector using 3D-printed parts and also factored in sufficient buffer time to account for delays in manufacturing. These steps ensured that we had a working manufactured end-effector three weeks before our final demonstration. Wherever possible, we also ordered spare components to account for unprecedented failures. Our last critical risk was bottlenecks with system integration. We believe our greatest asset as a team was our communication and teamwork. Throughout the project, each team member has been proactive about the development of their own sub-systems and has been cognizant of how their work would affect the rest of the system. In the fall semester, we also had more team knowledge transfer sessions, and team stand-ups to ensure everyone was aware of the progress in other sub-systems. A combination of these efforts ensured smooth final integration and a successful demonstration. We also ensured periodic and transparent communication with our sponsors, and their expertise helped us make the right decisions to mitigate our critical risks.

8.3.3 Improvements:

We believe that some of our risks were too broad in their scope. For instance, one of our main risks involved not meeting full-system performance requirements. This resulted in the mitigation strategies being too high-level and not specific enough for each sub-system. We realized that our mitigation strategies are more useful when they are specific actionable items rather than high-level guidelines. Many times during the spring semester, we failed to consistently keep our risks updated. This was especially true towards the end when the workload on each team member increased as we tried to meet our deadlines. However, realizing this shortcoming, we did a much better job at keeping our risks updated during the Fall 2022 semester.

Risk #	Risk	Type	Status	Risk Mitigation Actions
1	End effector development delays	Technical	**Mitigated**	• Start early and lock design by October 15th • 3D print to test before final manufacturing • Make system simple enough to get it manufactured in-house • Use 3D printed design as a fallback
2	Dynamic compensation not achievable	Technical	**Mitigated**	• Iterate with multiple control architectures • Utilize earlier compensation solution as a fallback • Evaluate need for isolating controller from ROS • Benchmark latencies in system • Use pub/sub communication instead of server/client communication
3	UI does not integrate with system	Technical	**Mitigated**	• Start working on the UI early • Plan architecture and consult each stakeholder • Start testing by Oct 31st
4	Performance requirements not met	Programmatic	**Mitigated**	• Track & evaluate quantification of performance requirements • Revisit performance requirements every sprint meeting • Have a risk-manager to track key risks
5	Integration issues between subsystems	Technical	**Mitigated**	• Define clear inputs and outputs of each subsystem • Host frequent meetings & retrospectives • Create documentation at the end of every milestone

Figure 29: Critical Project Risks

9 Conclusions

9.1 Lessons Learned

The MRSD capstone project was a key learning experience for all members of the team. From a technical perspective, we improved our programming skills in Python and C++, following industry standards and maintaining code modularity. Further, we made efforts to neatly document our code and progress. We realized the value of creating software architectures and flowcharts to organize data flow and functions in a complex system, like ours. Finally, we understood that it is important to iterate fast and fail quickly. During the Spring semester, we took a long time to pivot from some of our ideas — the Model Predictive Controller (MPC), poor hardware design, and so on. However, we realized this quickly and were much more flexible during Fall.

We learned that an ideal project management strategy may look different for different teams. Although we started off with JIRA as our task management and planning tool, we quickly pivoted to a simple whiteboard with sticky notes. This gave our team more visibility about pending tasks and boosted collaboration between team members. In addition, we learned that while planning timelines for deliverables, it is important to analyze potential risks and add buffer times to expected deadlines.

As a team, we learned the significance of clear and timely communication. Working together and conducting knowledge transfer sessions helped us leverage each others' diverse expertise. We made notable efforts to maintain a deep and professional relationship with our sponsors. Knowing their level of experience in the medical robotics domain, we insisted on bi-weekly 1-hour-long in-person meetings, where we discussed progress, brainstormed ideas, and received feedback and suggestions. We realize that periodic, in-person meetings are more efficient than virtual meetings.

9.2 Future Work

1. **User Interface:** Future work in the user interface will involve improving UI/UX design elements, adding cross-sectional views during pelvis-implant alignment, and adding more features for better surgical planning.

2. **Improving Simulation Environment:** Many changes that were made during implementation on the real hardware are not reflected in our simulation environment. We will update our simulation and possibly explore moving to MuJoCo for better controls simulation.

3. **Increasing Controller Frequency to 1 kHz:** As mentioned previously, one of the weaknesses of our current controller is that it is bottlenecked at 40 Hz. While it performs reasonably well, it would be much smoother and faster at tracking at a higher frequency, so we would like to use Kinova's lower-level API to implement a higher-frequency controller. This would also require us to move to a more real-time middleware such as ROS2, which does not have the same latency constraints as ROS1.

4. **Better camera:** We utilized an older version of the Atracsys Sprytrack Camera with a framerate bottlenecked at 55 Hz. In the future, we would like to use the next-generation version of the camera, such as the FusionTrack 500 which sports a better resolution and framerate of up to 300 Hz.

5. **Utilize New Robot Arm:** One next step we would definitely want to take would be moving away from using the Kinova Gen-3 as our robot arm as it is too compliant for our needs. Moving towards utilizing a more robust robot arm would allow us to improve our system's reaming accuracy.

6. **Overhauled Electrical System:** This semester we focused on developing a functional rather than a professional electrical system, leading us to move away from utilizing a PCB. In the future, we could move our microcontrollers, motor controllers, current sensors, and wiring onto one PCB board to simplify the electrical system. We noticed that our microcontroller sometimes misses encoder ticks from our motor encoders. To overcome this, we would like to move away from the Arduino Mega to a Teensy, which has a higher clock speed.

7. **Statistical Shape Modeling:** One of the key registration technologies that our sponsor already uses is something called Statistical Shape Modeling. Instead of requiring a CT scan of the pelvis before the surgery, which takes time and money, they allow the surgeon to "color in" the patient's pelvis with the registration probe. Using the collected point cloud, they then estimate the shape of the pelvis based on a learned model from thousands of previously collected scans of pelvises and use that estimated shape for registration. This makes registration faster, cheaper, and less error-prone, but requires more data before it can be effective.

8. **Clinical Testing and FDA Approval:** While we were able to prove that our technology works on a mock setup, we are yet to prove it to work on a real human body. To achieve this, the system will need to undergo rigorous cadaver testing and iteration through surgeon feedback. This will naturally pave the way to meet the necessary regulatory requirements before the system can enter the market as a product.

References

[1] B. Kayani, S. Konan, A. Ayuob, S. Ayyad, and F. S. Haddad, "The current role of robotics in total hip arthroplasty," *EFORT Open Reviews*, vol. 4, no. 11, p. 618–625, 2019.

[2] L. J. L. T. R. C. C. L. . Z. J. R. Lewinnek, G. E., "Dislocations after total hip-replacement arthroplasties," *The Journal of Bone Joint Surgery*, vol. 60, no. 2, p. 217–220, 1978.

[3] L. S. D. L. V. D. S. C. C. J. . M.-A. S. Rivière, C., "Spine–hip relations in patients with hip osteoarthritis," *EFORT Open Reviews*, vol. 3, no. 2, p. 39–44, 2018.

[4] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

10 Appendix

Full-Size Use-case Graphic

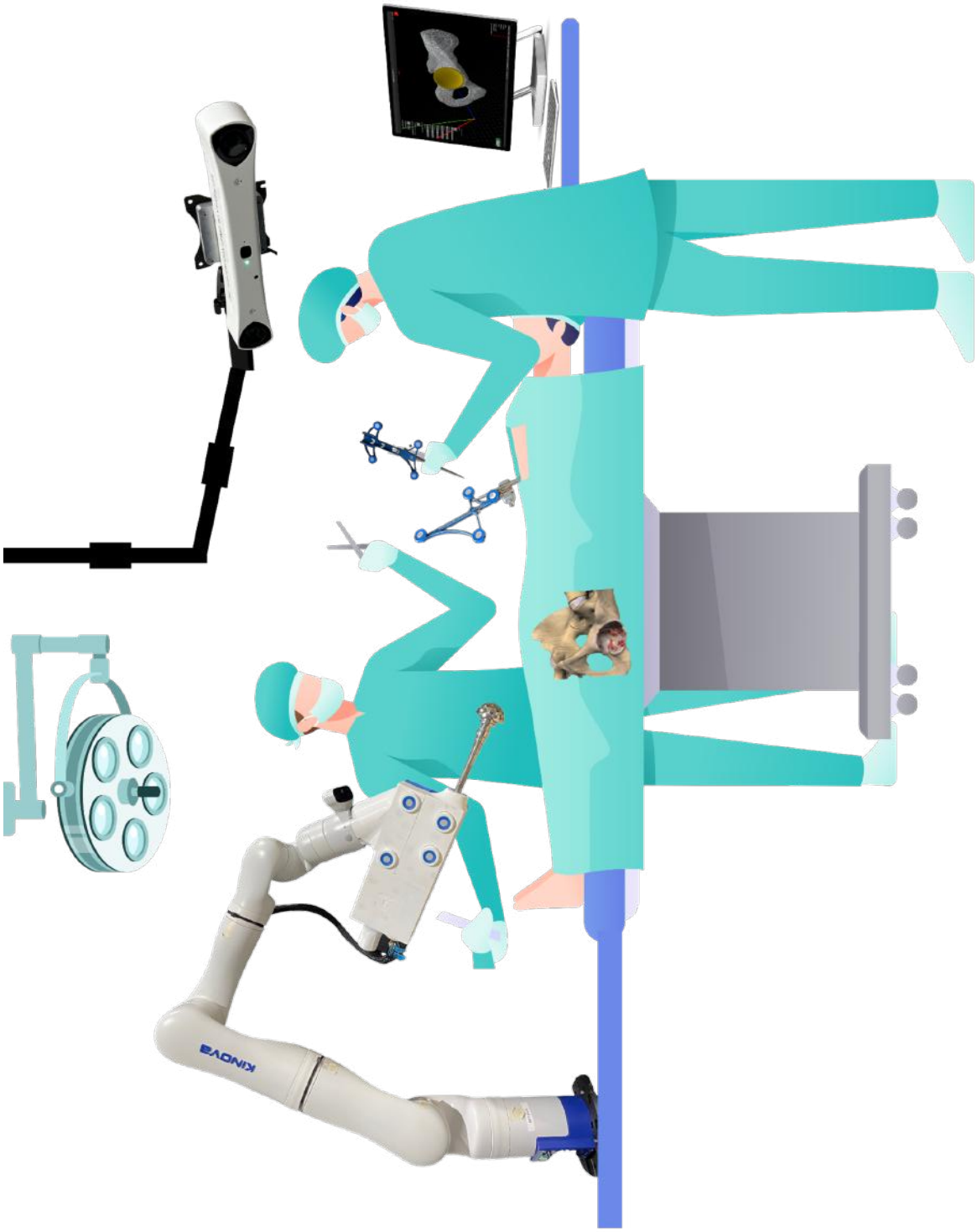

Figure 30: Full Size Use-case Graphic

Full-Size Functional Architecture

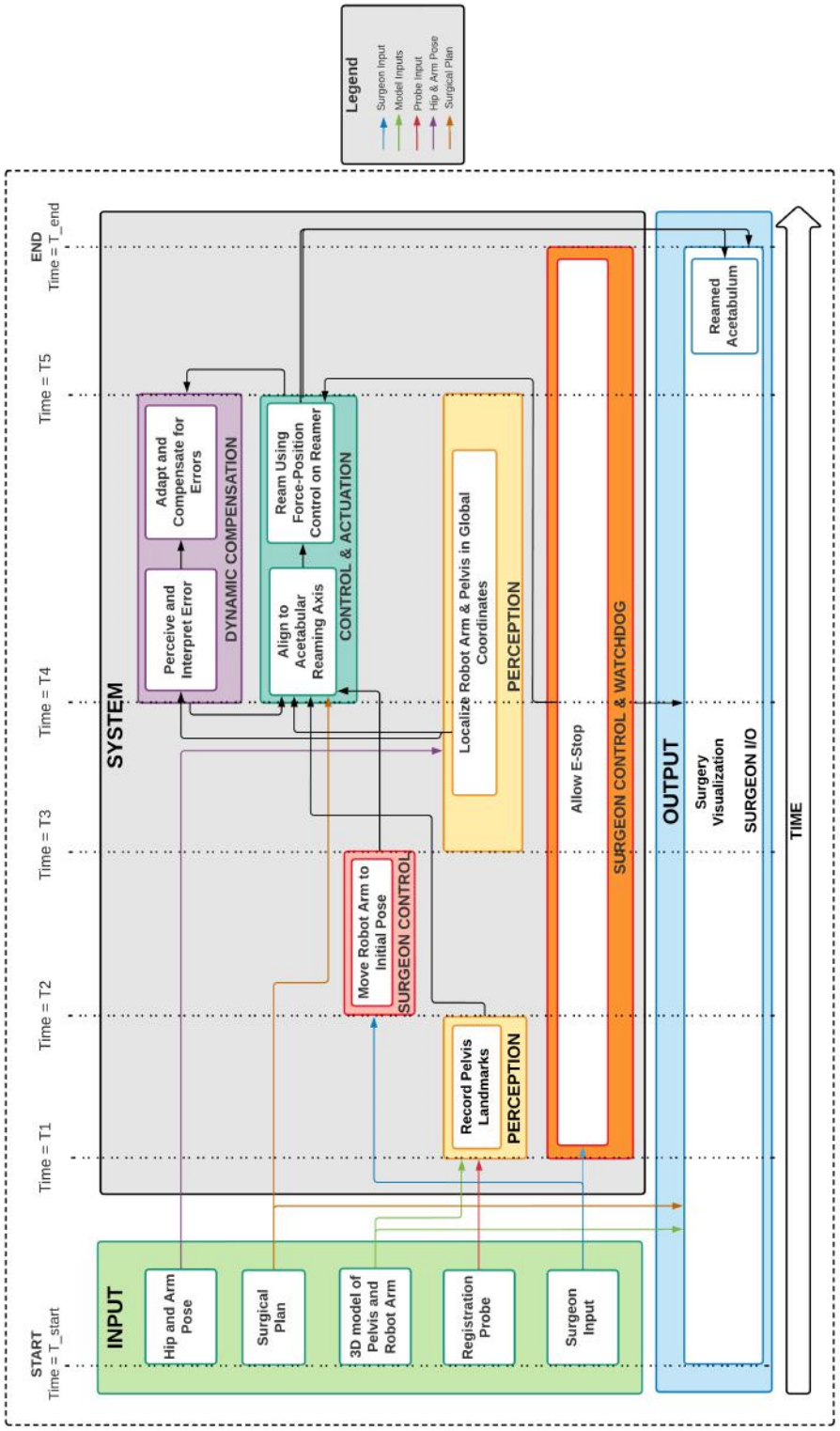

Figure 31: Full Size Functional Architecture : Revised

Full-Size Cyberphysical Architecture

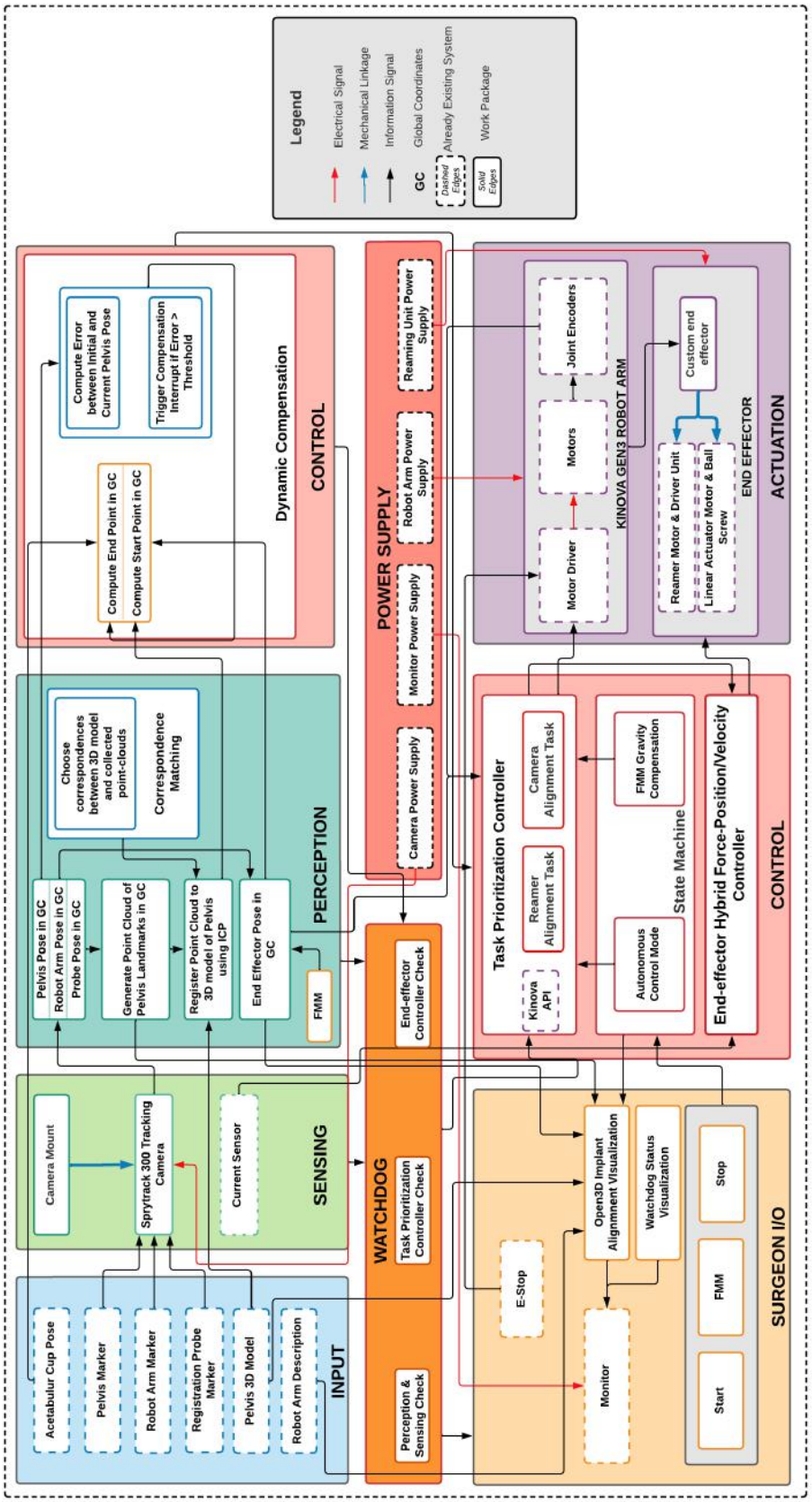

Figure 32: Full Size Cyberphysical Architecture

Full Size System Level Trade Study

Concept	Non-Robotic Hip Replacement	Computer-Guided Hip Replacement	Semi-Autonomous Robotic Hip Replacement	Fully-Autonomous Robotic Hip Replacement
Description	Reaming the acetabulum using no additional technology	Reaming the acetabulum with technology that allows a surgeon to visualize the inclination and anteversion they are reaming at	Reaming the acetabulum with a robotic arm which allows surgeons to use push on a reaming tool which is kept along a specific axis as per the planned location of the acetabular cup	Reaming the acetabulum with a robotic arm with no external surgeon input aside from a planned location of an acetabular cup
Evaluation Criteria	**Weighting Factor %**	**Value: 1 - 10 Ranging from *Inadequate* (1) to *Excellent* (10)**		
Surgical Time	10.00%	5	4	4	5
Feedback to Surgeon	15.00%	5	7	7	4
Achievable Accuracy & Patient Outcome	45.00%	5	6	8	10
Probability of System Failure	5.00%	5	8	5	8
Severity of System Failure	2.50%	5	5	3	2
Detectability of System Failure	2.50%	5	8	5	8
Longevity	7.00%	5	5	2	4
Time/Effort of Setup	7.00%	5	4	3	3
User Training	6.00%	5	4	2	4
Total:	**100.00%**	**5**	**5.745**	**5.97**	**6.98**

Figure 33: Full Size System Level Trade Study

Acetabular Alignment Task Steady-State Error

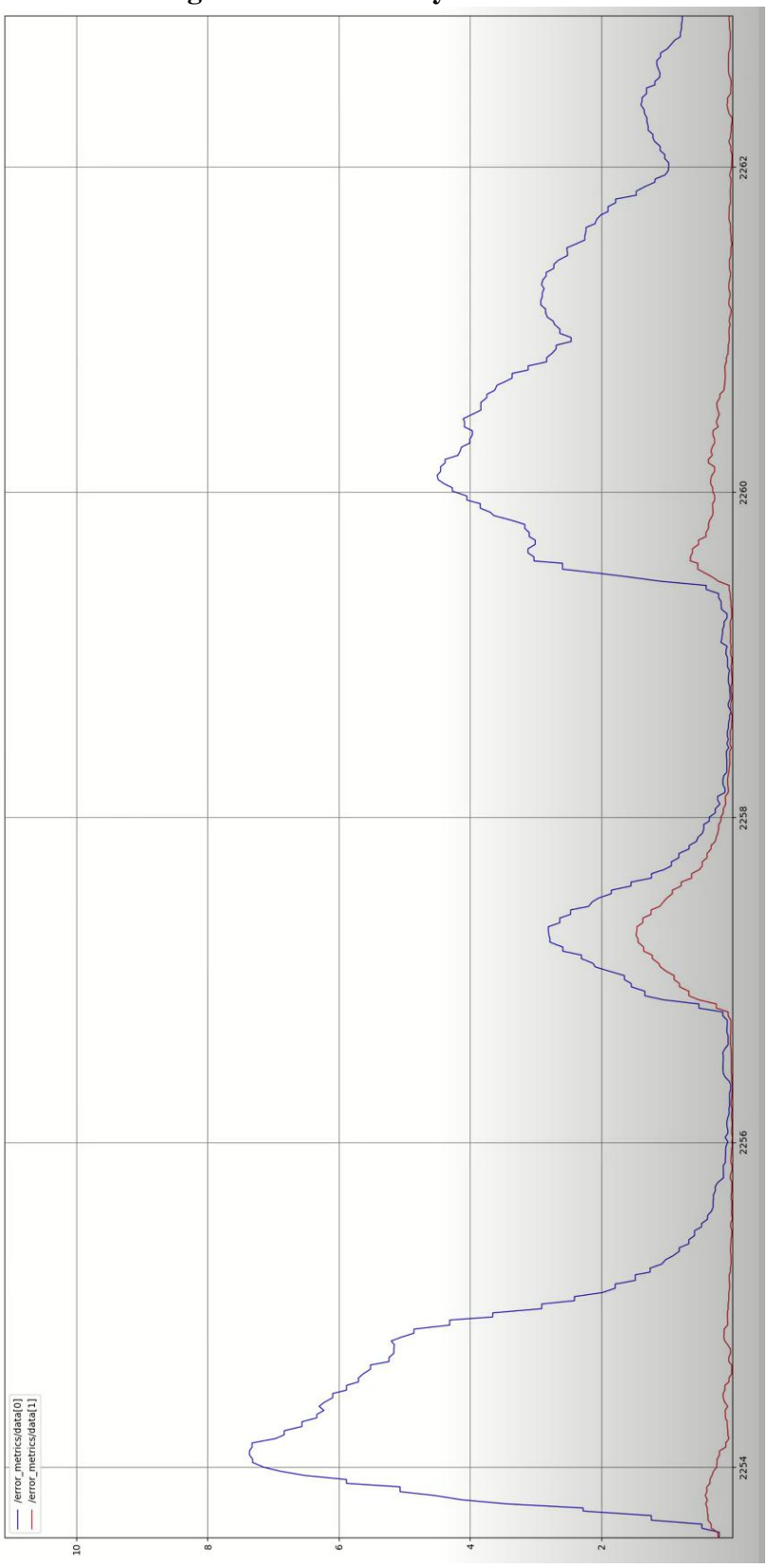

Figure 34: Steady-State Error of Tracking the Pelvis. Blue is Position Error, and Red is Orientation Error.

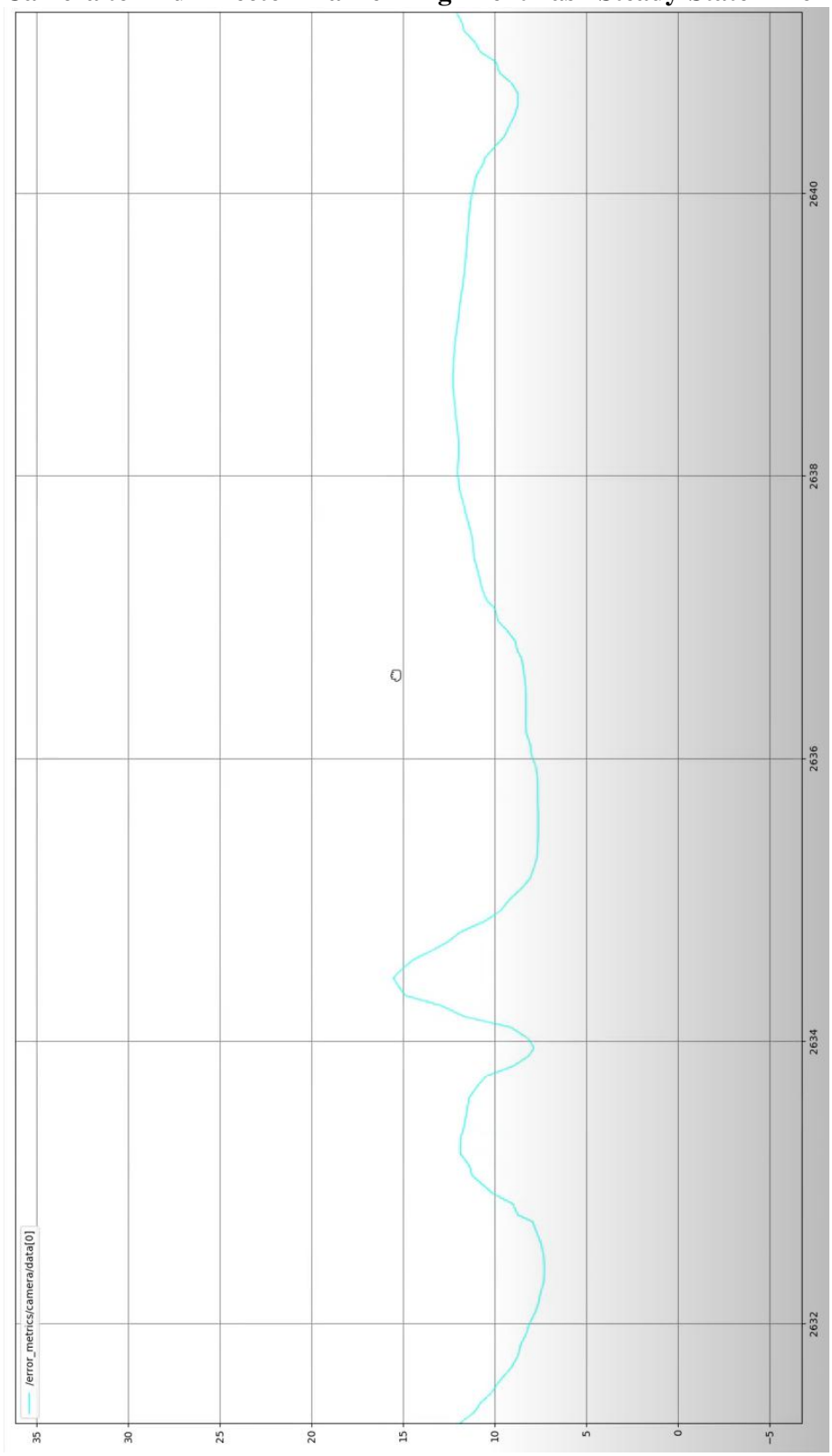

Figure 35: Steady-State Error of Tracking the End-Effector Markers to the Camera

Full-Size Project Fall Schedule

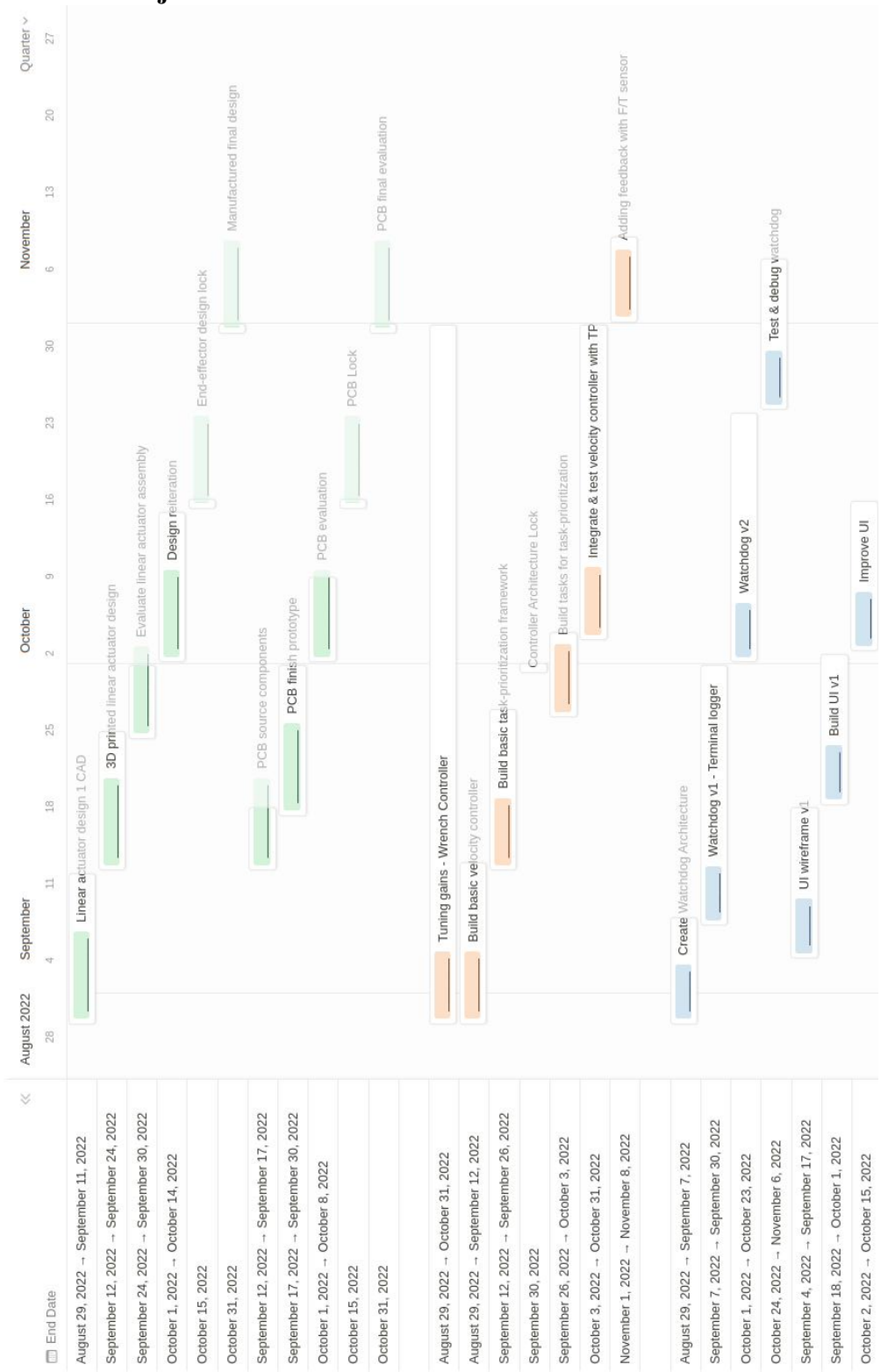

Figure 36: Full Size Project Schedule

Full-Size Project Risks Summary

Risk #	Risk	Type	Status	Risk Mitigation Actions
1	End effector development delays	Technical	**Mitigated**	• Start early and lock design by October 15th • 3D print to test before final manufacturing • Make system simple enough to get it manufactured in-house • Use 3D printed design as a fallback
2	Dynamic compensation not achievable	Technical	**Mitigated**	• Iterate with multiple control architectures • Utilize earlier compensation solution as a fallback • Evaluate need for isolating controller from ROS • Benchmark latencies in system • Use pub/sub communication instead of server/client communication
3	UI does not integrate with system	Technical	**Mitigated**	• Start working on the UI early • Plan architecture and consult each stakeholder • Start testing by Oct 31st
4	Performance requirements not met	Programmatic	**Mitigated**	• Track & evaluate quantification of performance requirements • Revisit performance requirements every sprint meeting • Have a risk-manager to track key risks
5	Integration issues between subsystems	Technical	**Mitigated**	• Define clear inputs and outputs of each subsystem • Host frequent meetings & retrospectives • Create documentation at the end of every milestone

Figure 37: Full Size Project Risks Summary