Critical Design Review Report

Autonomous Reaming for Total Hip Replacement

HIPSTER | ARTHuR

Team C:

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

May 5th, 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 increase 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 which 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 which can localize a pelvis to a robotic arm and plan a trajectory 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 to movements in the pelvis that may occur naturally during a reaming operation.

Contents

1 Project Description

Total Hip Arthroplasty (THA) Surgery as performed in present-day 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 of 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 the 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 the ARTHuR Robot (Autonomous Reaming for Total Hip Replacement Robot), which will maximize the accuracy of acetabular reaming, while remaining robust during surgery.

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 side such that the hip which is being operated on is facing upwards. Dr. Williams, using typical surgical technique 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 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, a cutting trajectory 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 trajectory 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 which can cut power to the robot.

Once the robot arm has completed the planned trajectory, it stops and allows for 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.

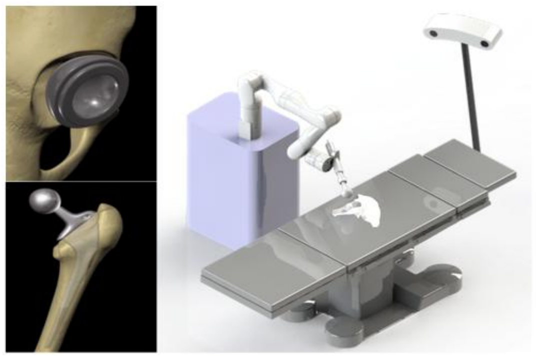


Figure 1: Use case graphical representation

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 localize the robot arm in real-time with respect to the pelvis before and during surgery	**M.P.1.1** The system will localize the robot arm in real-time with respect to the pelvis before and during surgery with a latency less than or equal to 50 ms	Latency of Atracsys Sprytrack 300 is less than 25ms; Processing time about 25 ms
	M.P.1.2.1 The system will localize the robot arm in real-time with respect to the pelvis before and during surgery with a position error of less than 1 mm	Survey sent to surgeons and literature review suggest a desired position error of less than 2 mm [4]. Combining **M.P.1.2.1** and **M.P.3.1** will result in a combined position error of less than 2 mm.
	M.P.1.2.2 The system will localize the robot arm in real-time with respect to the pelvis before and during surgery with an orientation error of less than 1.5-degrees	Survey sent to surgeons and literature review suggest a desired orientation error of less than 3-degrees [4]. Combining **M.P.1.2.2** and **M.P.3.2** will result in a combined orientation error of less than 3-degrees.
M.F.2 The system shall plan the trajectory of the robot arm based on the given surgical plan	**M.P.2** The system will plan the trajectory of the robot arm based on the given surgical plan with a latency less than or equal to 150 ms	Total latency of the system should be less than 500 ms.
M.F.3 The system shall execute surgical plan by reaming along the generated trajectory	**M.P.3.1** The system will execute surgical plan by reaming along the generated trajectory with an position error of less than 3 mm	Survey sent to surgeons and literature review suggest a desired position error of less than 2 mm [4]. Combining **M.P.1.2.1** and **M.P.3.1** will result in a combined position error of less than 2 mm.
	M.P.3.2 The system will execute surgical plan by reaming along the generated trajectory with an orientation error of less than 3-degrees	Survey sent to surgeons and literature review suggest a desired orientation error of less than 3-degrees [4]. Combining **M.P.1.2.2** and **M.P.3.2** will result in a combined orientation error of less than 3-degrees.
M.F.4 The system shall compute error and interpret the movement of the pelvis during reaming	**M.P.4.1** The system will compute error and interpret the movement of the pelvis during reaming with a latency less than or equal to 50 ms	Latency similar to localization
	M.P.4.2 The system will generate a new trajectory if the interpreted position and orientation errors are greater than 1 mm or greater than 1.5-degrees.	Survey sent to surgeons and literature review suggest a desired position and orientation error of less than 2 mm and 3-degrees [4]. Therefore, the thresholds for compensating for these errors should be less than these desired errors.
M.F.5 The system shall adapt and compensate for movement by generating a new trajectory	**M.P.5** The system will adapt and compensate for movement by generating a new trajectory with a latency less than or equal to 150 ms	Latency similar to trajectory planning
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 by keeping path to be planned short
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 tele-surgery, latency 150 ms is found to be noticeable to surgeons, and degrades 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

As shown in Table 1, are our mandatory functional and performance requirements. Since the CoDRR, most of the requirements remain the same. The only requirements that have changed since the CoDRR are **M.P.3.1** **M.P.3.2**. Originally, these requirements were to have an execution error of no more than 1 mm and 1.5 degrees. However, after extensive testing on our given hardware, our arm, the Kinova Gen 3, does not have enough holding torque and stiffness to dampen the vibrations from reaming, which will inevitably introduce more error into our execution. Therefore, we have relaxed those requirements to reflect that.

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 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 as 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 indicates that both happen in tandem. The **inputs** are as follows:

1. The hip and arm positions inferred by the marker positions using the camera.
2. The 3D model of the the pelvis of the patient obtained pre-operatively, such as using a CT-Scan.
3. 3D model of the robot arm.
4. The surgical plan provided in the form of the pose of the acetabular implant in 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.

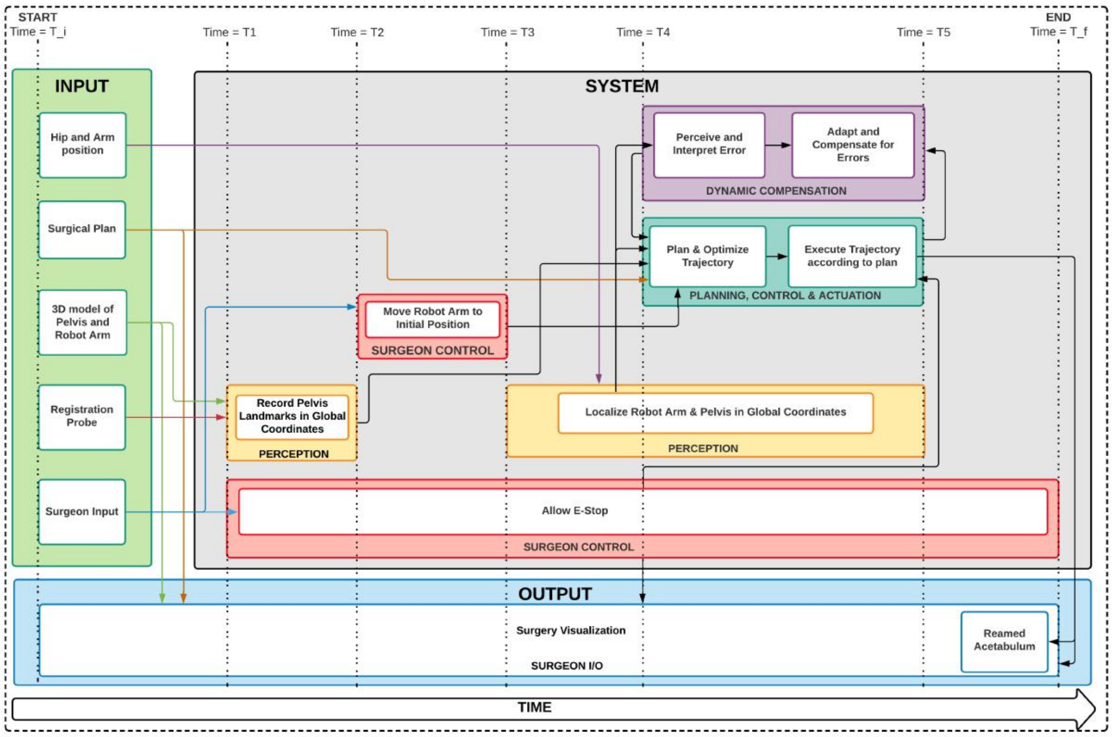

START
Time = T_i
Time = T1
Time = T2
Time = T3
Time = T4
Time = T5
END
Time = T_f
INPUT
Hip and Arm position
Surgical Plan
3D model of Pelvis and Robot Arm
Registration Probe
Surgeon Input
SYSTEM
Perceive and Interpret Error
Adapt and Compensate for Errors
DYNAMIC COMPENSATION
Plan & Optimize Trajectory
Execute Trajectory according to plan
PLANNING, CONTROL & ACTUATION
Move Robot Arm to Initial Position
SURGEON CONTROL
Record Pelvis Landmarks in Global Coordinates
PERCEPTION
Localize Robot Arm & Pelvis in Global Coordinates
PERCEPTION
Allow E-Stop
SURGEON CONTROL
OUTPUT
Surgery Visualization
SURGEON I/O
Reamed Acetabulum
TIME

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 joint encoders. The end-effector marker is used as a calibration target to perform hand-eye calibration prior to the surgery. The planning, 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. Based on the start and end points computed by the planning block, an optimized trajectory is determined and executed. Simultaneously, the dynamic compensation module ensures that the end point is recomputed in the event that the pelvis moves from its initial position upon being impinged by the reaming tool.

Finally, the **output** is the reamed acetabulum as per the accuracy defined by our performance requirements. Throughout this process, the various steps and current status of the surgery can be visualized on a monitor which formed the surgeon I/O component of our system. An enlarged version of this functional architecture can be found in the appendix in figure 30.

5 Cyberphysical Architecture

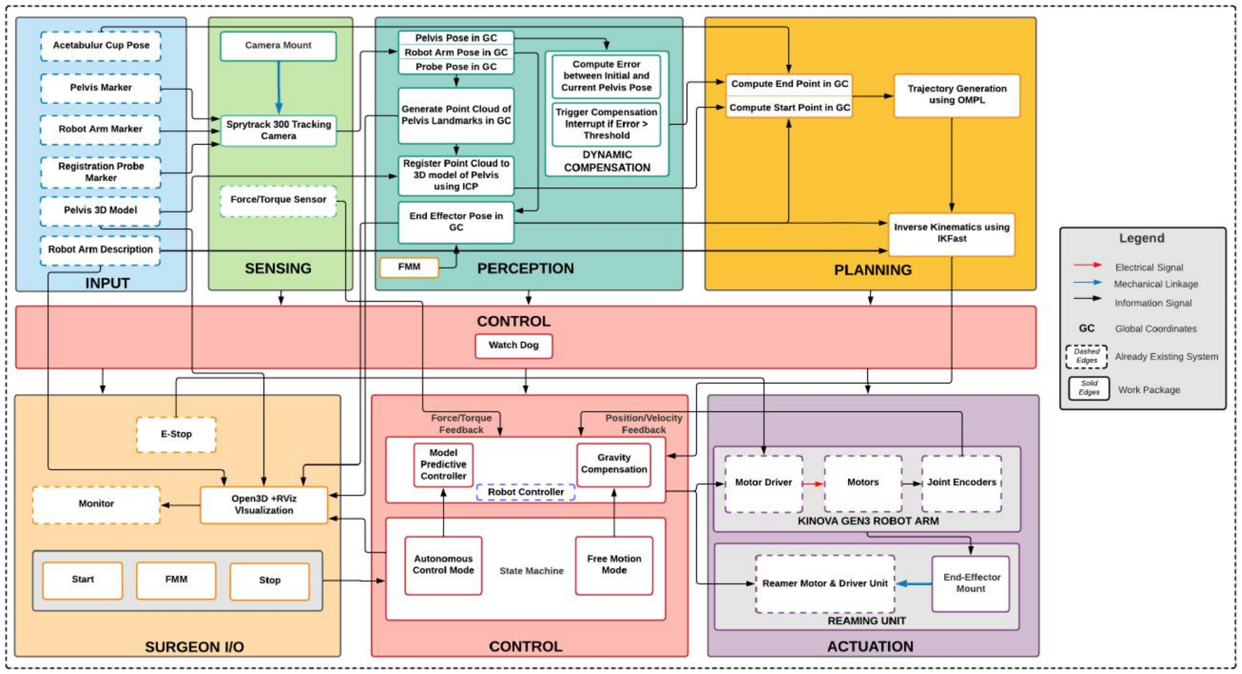


Figure 3: Cyberphysical Architecture

Figure 3 below shows the cyberphysical architecture for our system. The major components of the cyberphysical architecture are derived from the functional architecture. On a high-level, the various block are inputs, sensing, perception, planning, control, actuation and surgeon I/O. We also have the power-supply for each module, and a watch-dog that ensures that the various sub-modules are functioning as expected. Any block here with a dashed border such as those seen in the input block do not directly translate to a work-package, and those with a solid border will contribute to a work-package in the work breakdown structure. 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 on the pelvis, robot arm and the registration probe respectively. The surgical plan in the form of the acetabular cup pose is used to compute the end-pose. 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.

3. **Perception**: Within the perception block, the sensor data from the camera is used to determine the pose of the patient's pelvis, robot arm and registration marker in robot base frame of reference. Currently, the pose of the end effector pose determined using the encoder data from the robot's joints - while the marker is used as a calibration target. A hand-eye calibration procedure prior to surgery will localize the robot arm's base frame with respect to the camera. A one time process of landmark acquisition happens at the beginning of each surgery where key landmarks from the patients 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. The dynamic compensation module is used to continuously monitor the error between the current pelvis pose and the initial pelvis pose. Should this error exceed a predefined threshold, an interrupt is triggered to ensure that the planning module compensates for this movement and recomputes a new trajectory for the controls module to execute.

4. **Planning**: The start and end points of reaming is computed based on the surgeon input and the surgical plan respectively. Using this, the reaming trajectory to be followed is generated using the Pilz Industrial Motion Planner. The optimized trajectory is then used an input to the IKFast inverse kinematics plugin which returns the desired joint angles for the joint to execute at every time-step. If a compensation interrupt is triggered from the perception block, the end point of reaming is recomputed to compensate for the motion of the pelvis during surgery.

5. **Control**: The control module is composed of two modes which will be set by the Surgeon I/O. A state machine is used switch between Autonomous Control Mode or Free Motion Mode (FMM). If the state machine is in Autonomous Control Mode, then the control loop will take inputs from the planned trajectory and use those inputs in the control loop. If the state machine is in Free Motion Mode, then the control loop will be using gravity compensation to keep the robot in place unless moved manually by the surgeon. A watch-dog module continuously monitors and takes overriding controls actions should there be any sub-system level malfunctions.

6. **Actuation**: The control loop sends velocity commands to the micro-controller through a ROS topic. The microcontroller continuously subscribes to this command, and using PID velocity control, commands the desired PWM values required to drive the motors. The joint positions from the encoders will be constantly monitored as feedback to the control loop. The reaming unit in the actuation block will be controlling the spindle rate and torque of the reamer during the reaming operation. The reamer is mounted to the robot arm mechanically using a customized reamer adapter tool.

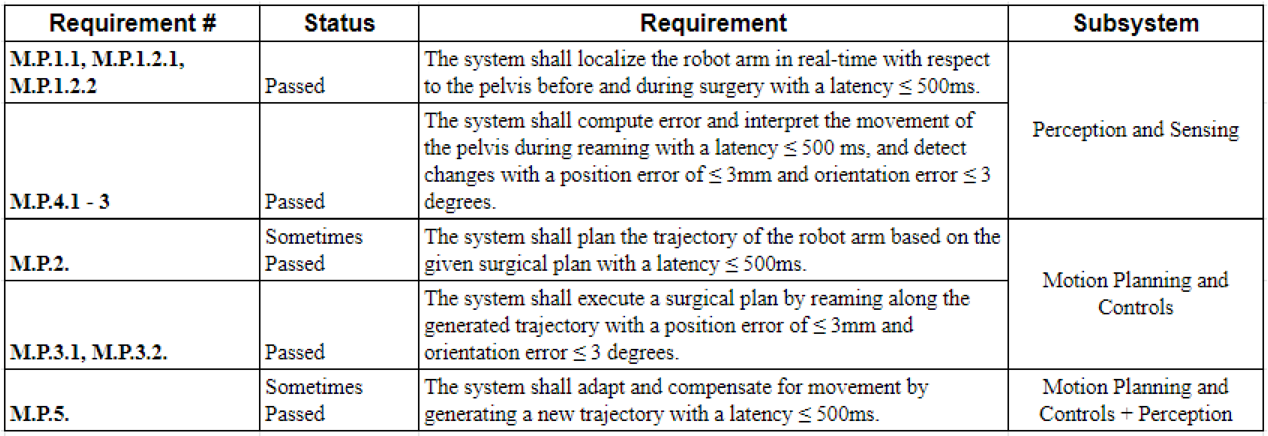
7. **Surgeon I/O**: The surgeon would be provided with a visualization that they can monitor throughout the procedure. A control interface was also included to allow the surgeon to control the operating mode of the robot through the state machine in the control module. An E-Stop will be available to them at all times for stopping the robot arm at any time during the procedure.

6 Current System Status

6.1 Spring-Semester Targeted System Requirements

For the Spring Semester, the team had the following set of requirements shown in Table 3. Some of these requirements are more relaxed compared to our system-level requirements in section 3, such as the latency requirements for localizing the arm and pelvis being relaxed to 500 ms instead of 50 ms, latency requirements for motion planning to generate a trajectory within 500 ms instead of 150 ms.

Table 3: Targeted System-Level Requirements for Spring Semester



Requirement #	Status	Requirement	Subsystem
M.P.1.1, M.P.1.2.1, M.P.1.2.2	Passed	The system shall localize the robot arm in real-time with respect to the pelvis before and during surgery with a latency ≤ 500ms.	Perception and Sensing
M.P.4.1 - 3	Passed	The system shall compute error and interpret the movement of the pelvis during reaming with a latency ≤ 500 ms, and detect changes with a position error of ≤ 3mm and orientation error ≤ 3 degrees.	
M.P.2.	Sometimes Passed	The system shall plan the trajectory of the robot arm based on the given surgical plan with a latency ≤ 500ms.	Motion Planning and Controls
M.P.3.1, M.P.3.2.	Passed	The system shall execute a surgical plan by reaming along the generated trajectory with a position error of ≤ 3mm and orientation error ≤ 3 degrees.	
M.P.5.	Sometimes Passed	The system shall adapt and compensate for movement by generating a new trajectory with a latency ≤ 500ms.	Motion Planning and Controls + Perception

6.2 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 current status of our system.

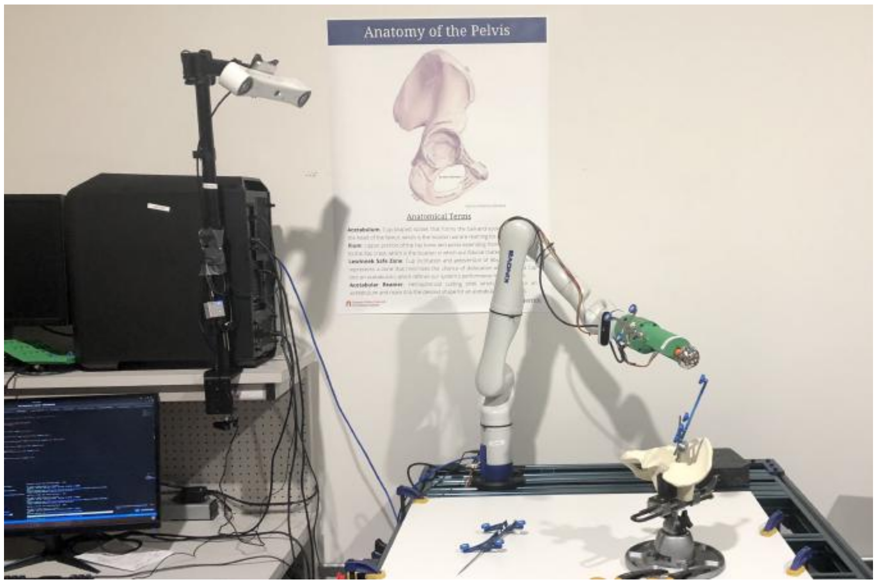


Figure 4: Overall System Depiction

Starting on the left side of figure 4 is our computer which is running ROS Noetic and handles all of our data processing and planning. Above the computer is our Atracsys Sprytrack 300 camera, which is composed of two cameras which 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. Connected to the computer are several different USB and Ethernet cables which are used for a variety of reasons, such as interfacing with the robotic arm and reamer motor, and collecting data from the system via integrated sensors.

Our base of our workspace is a Vention table which we thankfully received from Professor Kroemer. We added boards to the top and bottom of our workspaces, one for ease in working with the arm and the surgical setup on top, and one for placing all of our electrical components on the bottom. Our robot arm is a Kinova Gen-3 and it was placed towards the back left of the Vention table to provide it with room to plan towards the pelvis. Attached to the end-effector of the Gen-3 arm is our custom reaming end-effector, which is comprised of a 3D printed adapter to the force torque sensor, an ATI Axia-80 M20 force torque sensor, a 3D printed motor adapter, a Servo City planetary gear motor with encoders and a face plate, a fiducial IR marker, a 3D printed reamer head adapter, and a reamer head. Also on top of the workspace is a Panavise vise which is holding a Sawbone pelvis, with the acetabulum of the pelvis aiming towards the base of the robot arm. The Sawbone pelvis also has a fiducial IR marker attached to it via a bone screw and an adapter.

On the bottom of our workspace, which can be seen in figure 5, is where we kept all our electrical components. Down there is our custom PCB for controlling the reamer motor, a power supply for the motor and the force-torque sensor, the power adapter for the Kinova Gen-3 arm, and an ethernet splitter so that we can communicate with the arm and with the force-torque sensor.

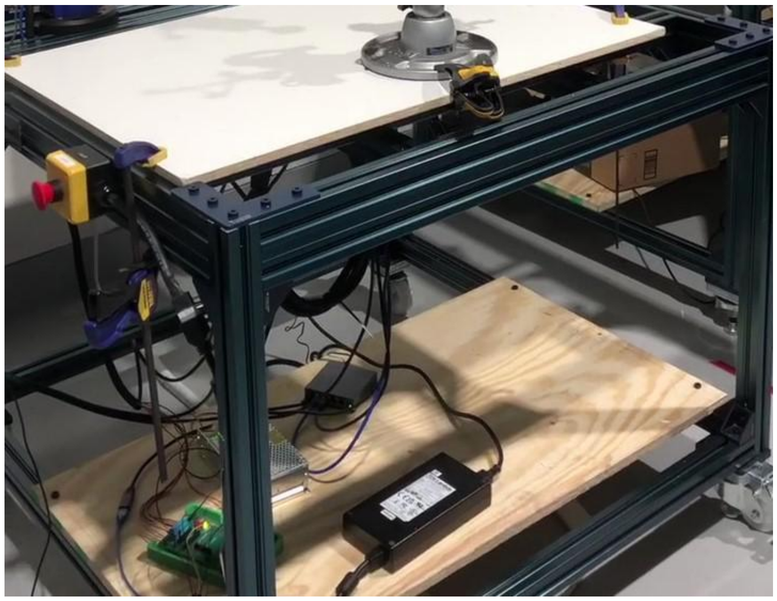


Figure 5: Electrical Setup Under Table

6.3 Subsystem descriptions/depictions

6.3.1 Inputs

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

1. Pelvis Marker: A set of fiducial markers that allows the camera to track the pose of the pelvis in the surgical setting.
2. End-Effector Marker: A set of fiducial markers that allows the camera to track the pose of the robot arm end-effector. Currently, we are only using this as a calibration target for hand-eye calibration.
3. Registration Probe: A set of fiducial markers mounted on a probe with a 1 mm radius tip than can be used to collect a pointcloud 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 planning and controls sub-systems to execute the surgical plan.

The various inputs into the system are summarized in Figure 6.

6.3.2 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

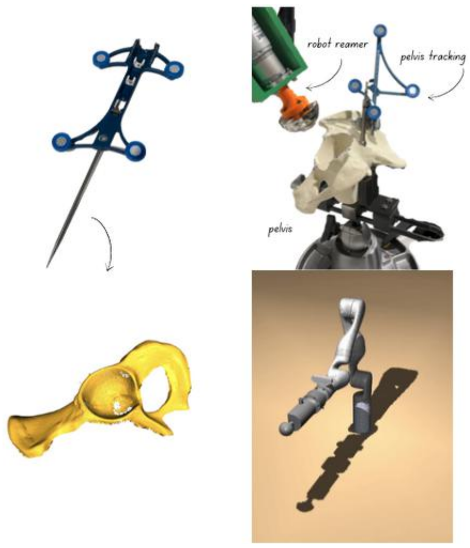


Figure 6: The various system inputs (i) Registration Probe (ii) Pelvis EE Marker (iii) Pelvis CAD model (iv) Robot Arm CAD model

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 FPS. 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 the Figures 7 and 8.

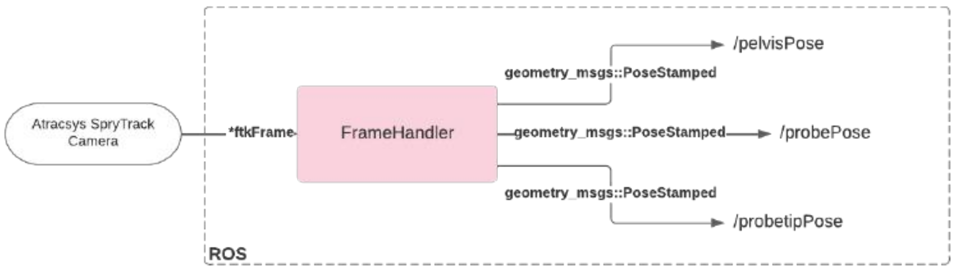

Atracsys SpryTrack Camera
*ftkFrame
FrameHandler
geometry_msgs::PoseStamped
/pelvisPose
geometry_msgs::PoseStamped
/probePose
geometry_msgs::PoseStamped
/probetipPose
ROS

Figure 7: Frame Handler

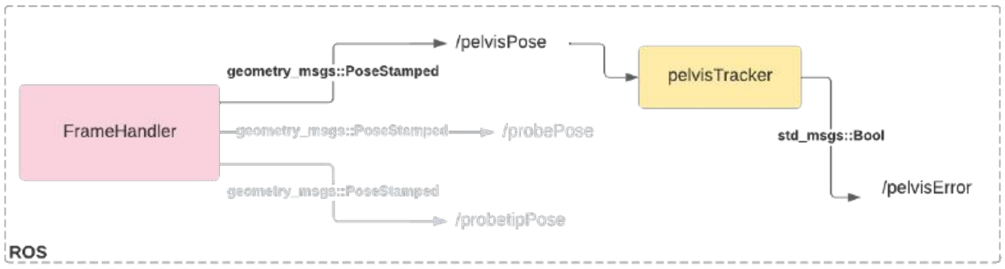


Figure 8: Pelvis Tracker

Pointcloud Collection and Registration: As an initial calibration step, the perception subsystem allows the surgeon to collect both a sparse and dense pointcloud. Once this pointcloud is collected, the surgeon can match landmark correspondences between the preoperative scan and collected pointcloud. 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 pointcloud with selected landmark points is shown in 9. This enables an initial guess for the registration process. Using the Iterative Closest Point (ICP) algorithm, the transformation between the pelvis pointcloud and pelvis scan is obtained. A sample registered pointcloud can be seen in Figure 10.

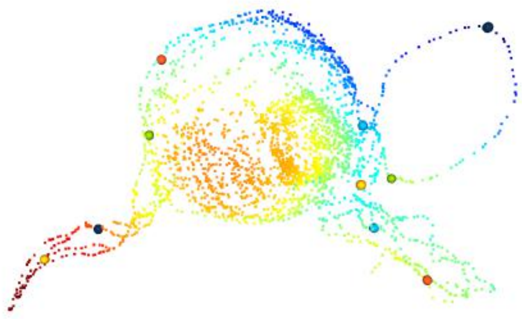


Figure 9: Collected Pointcloud Visualized in Open3D

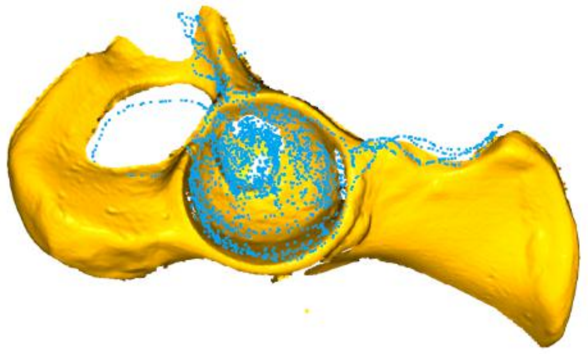


Figure 10: Results of Pointcloud Registration

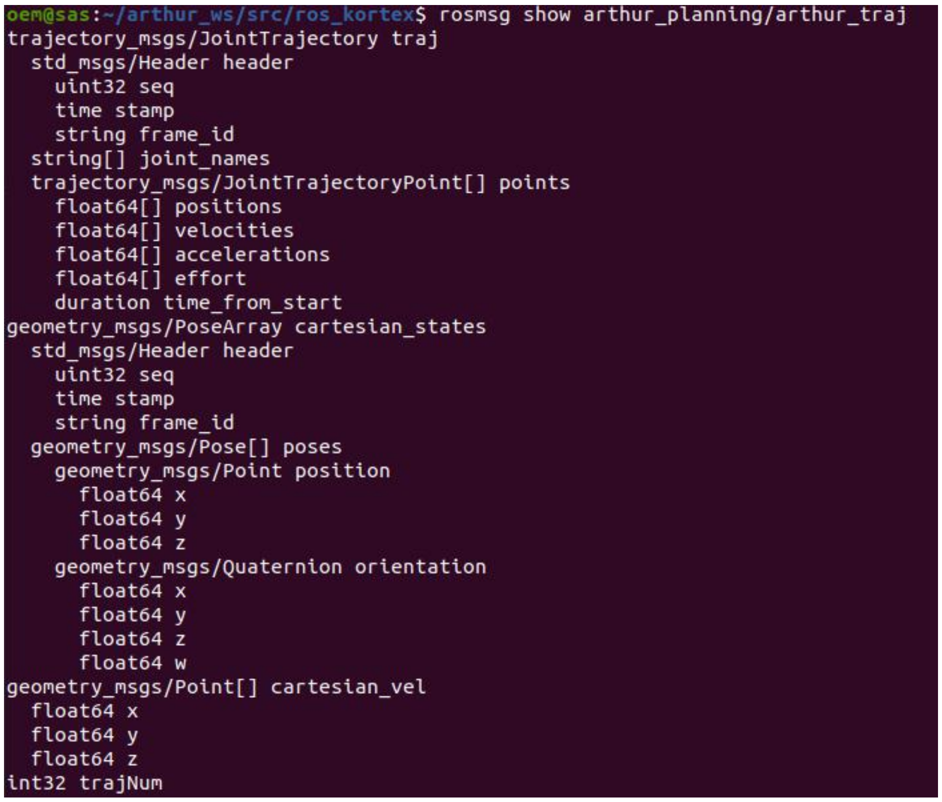
6.3.3 Planning

The planning node is responsible for generating linear trajectories between the current pose of the robot and the reaming end point. Using the free motion mode, the surgeon is expected to bring the tool closer to the site of surgery, after which the controller aligns the orientation of the end-effector with the reaming end point. The planning node operates after this realignment and passes the trajectory generated to the controller.

For planning, we use the Pilz industrial motion planner within MoveIt!. The Pilz planner essentially works as a trajectory generator and provides analytical solutions for the requested trajectory. The planner is successful in providing repeatable and deterministic solutions. The planner is also complemented by IKFast, which is an analytical inverse kinematics solver. The plan generated for our ARTHuR system is with respect to the robot base.

To transfer the information of the generated trajectory, we created a custom ROS message type as seen in Fig. 11. The message consists of the joint trajectory message, pose array, point and an int32. The joint trajectory message holds information of the joint positions, velocities and accelerations at each waypoint in the trajectory. The pose array holds the cartesian states of the end-effector, namely the 3D position and orientation (in quaternion) of the tool tip. The list of point messages holds the cartesian velocities at each waypoint. Finally, we also have an integer variable called ’trajNum’ which is used to keep track of the number of trajectories generated from the beginning and helps in dynamic compensation.

The joint state information could be directly accessed using MoveIt! function calls. The cartesian states had to be computed by calculating forward kinematics of the joint positions at each waypoint. The cartesian velocities were calculated using simple arithmetic i.e. dividing the difference of two consecutive cartesian positions by the time period. The Pilz planner is forced to plan a linear path in cartesian space while satisfying the joint limits specified by Kinova.



```
oem@sas:~/arthur_ws/src/ros_kortex$ rosmsg show arthur_planning/arthur_traj
trajectory_msgs/JointTrajectory traj
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  string[] joint_names
  trajectory_msgs/JointTrajectoryPoint[] points
    float64[] positions
    float64[] velocities
    float64[] accelerations
    float64[] effort
    duration time_from_start
geometry_msgs/PoseArray cartesian_states
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  geometry_msgs/Pose[] poses
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
geometry_msgs/Point[] cartesian_vel
  float64 x
  float64 y
  float64 z
int32 trajNum
```

Figure 11: ARTHuR custom message type

Whilst the system is in dynamic compensation mode, the controller realigns the end-effector to the new position of the pelvis. After this, a flag is sent to the planner to re-plan the trajectory from the robot's current pose. At this point, the new current position is used to again plan a linear path to the new reaming end point in space.

6.3.4 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 executes the trajectory given by the planning subsystem, and Dynamic Compensation which adjusts and aligns the end-effector's pose with the pelvis's pose if the pelvis moves during the reaming operation.

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 their API calls. This made integration into our system fast and easy.

For Autonomous Control Mode, the controls subsystem is currently taking advantage of Kinova's Wrench Command API built into their system. Essentially, our subsystem can send wrench commands to the arm, which moves the arm with respect to chosen frames. For our system, we are using a mixed frame, where the forces we send in the wrench command are with respect to the base frame coordinates, while the torques we send in the wrench command are with respect to the end-effector frame. The trajectory given by the motion planner has several waypoints containing desired positions and orientations for the end-effector to move through. Using this framework, a PID controller was developed which takes the error between the current pose and desired pose as the input into the PID controller, and outputs the desired wrench to reach that desired pose. The output of the PID controller is then clamped at a max force of 20 Newtons and max torque of 8 Nm. This allows the controller to be more safe around humans as the wrench controller will only exert these maximum forces in worst-case scenario. Figure 12 shows the general flow diagram of how the Autonomous Control Mode moves through different states, and Figure 13 shows the control block diagram of the wrench controller, which takes the desired pose as the input (determined by the planner, camera, and surgical plan), and takes the actual pose of the end-effector as feedback.

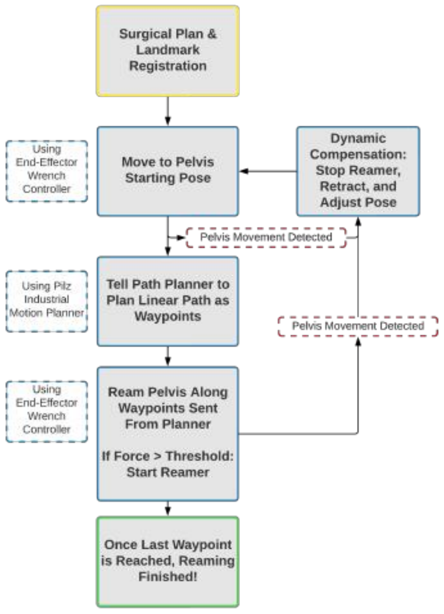

Surgical Plan & Landmark Registration
Using End-Effector Wrench Controller
Move to Pelvis Starting Pose
Dynamic Compensation: Stop Reamer, Retract, and Adjust Pose
Pelvis Movement Detected
Using Pilz Industrial Motion Planner
Tell Path Planner to Plan Linear Path as Waypoints
Pelvis Movement Detected
Using End-Effector Wrench Controller
Ream Pelvis Along Waypoints Sent From Planner
If Force > Threshold: Start Reamer
Once Last Waypoint is Reached, Reaming Finished!

Figure 12: State Machine of the Autonomous Control Mode

Because the motion planner can only plan from the current position of the end-effector, the controller has to first align its orientation with the reaming endpoint frame (usually determined in the surgical plan pre-surgery) and moves to a point 5 cm axially from the reaming endpoint frame. once it is properly aligned at the reaming start point (Figure 14), the controller tells the planner to generate a trajectory and then proceeds to move along the trajectory using the PID controller described earlier. Once the reamer makes contact with the acetabulum and exerts a force above a tuned threshold (detected by the force/torque sensor), the controller communicates with the reaming node to start the reamer (Figure 15). The reason to start reaming only after making full contact with the acetabulum is that reaming before making full contact can introduce high amplitude vibrations into the system, reducing accuracy. Waiting for full contact will allow reaction forces from the bone to cancel out, reducing vibrations. Once the end-effector reached all waypoints, the reaming operation is finished and the arm retracts.

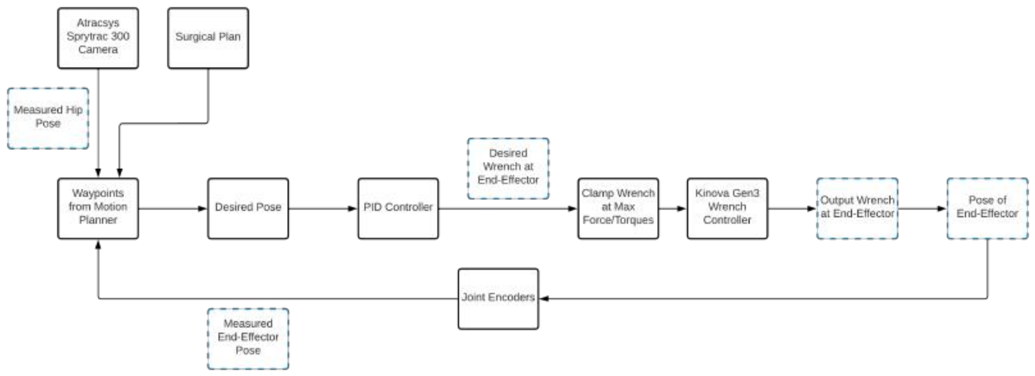

Atracsys Sprytrac 300 Camera
Surgical Plan
Measured Hip Pose
Waypoints from Motion Planner
Desired Pose
PID Controller
Desired Wrench at End-Effector
Clamp Wrench at Max Force/Torques
Kinova Gen3 Wrench Controller
Output Wrench at End-Effector
Pose of End-Effector
Joint Encoders
Measured End-Effector Pose

Figure 13: Control Block Diagram of the Wrench Controller

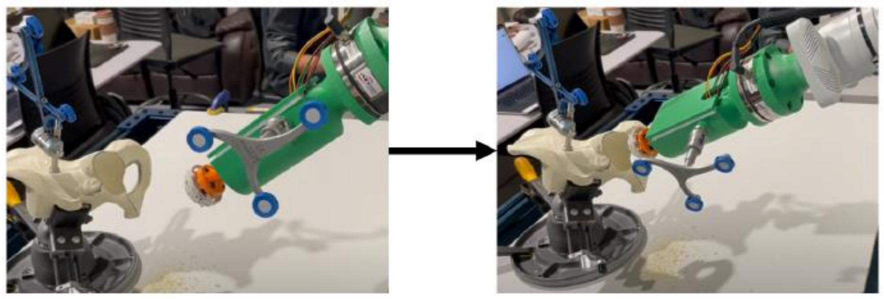


Figure 14: Reamer Aligning to Reaming Start Point

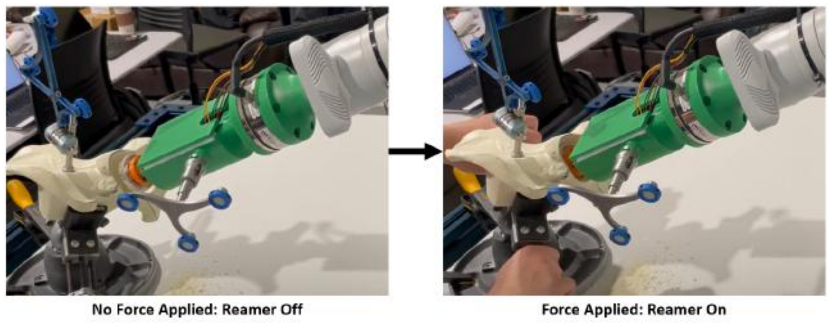


Figure 15: Reamer Starting After Force Applied on Acetabulum

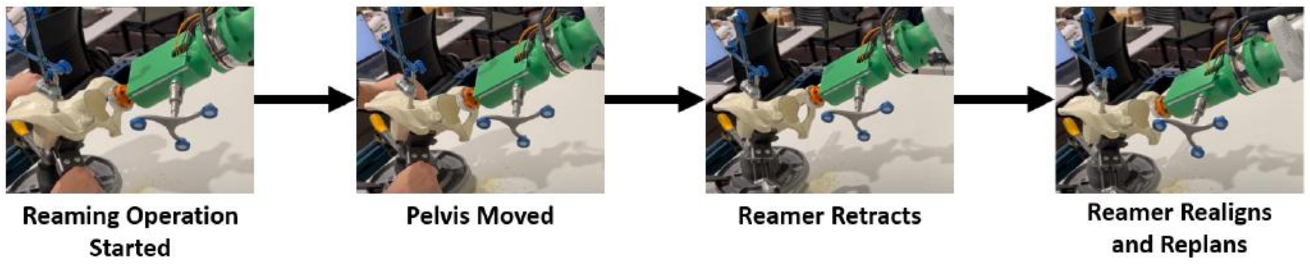


Figure 16: Dynamic Compensation

For Dynamic Compensation, the perception system will notify the controls subsystem when the pelvis pose has changed beyond a specific threshold as specified by our requirements. When dynamic compensation is triggered, the arm retracts 10 cm along the axis of reaming, and then reorients itself to the reaming start point as described above. The controller is constantly keeping track of 10 cm behind its current end-effector pose, so that when dynamic compensation occurs, it knows where to retract to. After reorienting itself, it then gets a new trajectory and begins reaming again. Dynamic Compensation is demonstrated in Figure 16.

6.3.5 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 creating the end-effector, creating a motor controller PCB, and setting up the force-torque sensor to communicate with ROS.

The reaming end-effector design went through several evolutions throughout the project, as can be seen in figure 17. Originally we thought to clamp around the connection to the fiducial marker on the reamer handle, but we were advised by our sponsor to instead try to connect around the ridges of the reamer handle. We designed and 3D printed this design, but upon integration we determined that the design was too long which caused some undesirable wobbliness in the system. Based on this, we decided to stop trying to implement the reamer handle in our design, and instead create a 3D printed design which mimics the geometry of the reamer handle.

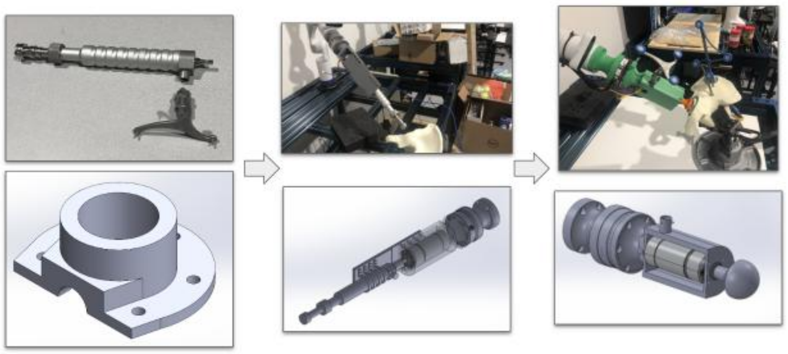


Figure 17: Evolution of the End-Effector

This led to our finalized design which can be seen in figure 18. This design featured three 3D printed parts, an adapter between the end-effector and the force-torque sensor, and a motor mount can hold a motor and mimic important geometries from the reamer handle, and a reamer head adapter. This final reaming end-effector worked perfectly for what we needed, as the assembly vibrated minimally during the procedure (most vibrations were through the entire arm) and the system was able to hold up to the torques during and operation. While we would like to pursue potential improvements to this system in the future, we are satisfied overall with this design.

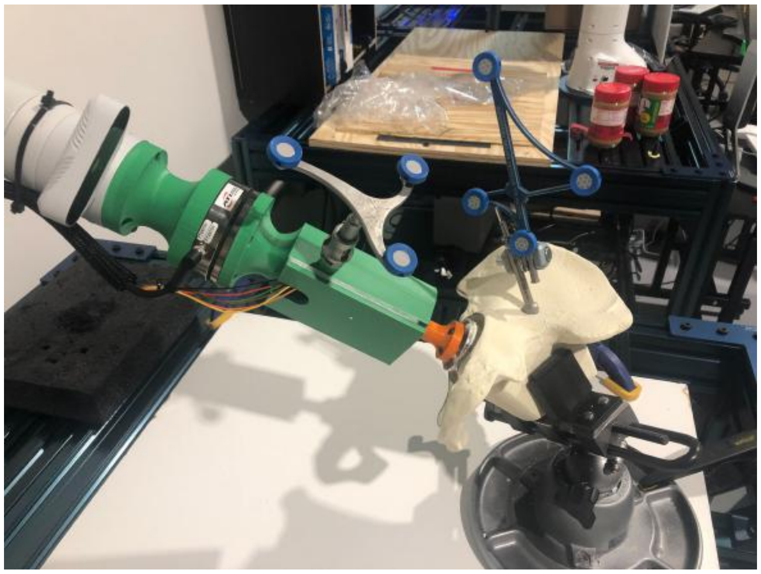


Figure 18: Final Reaming End-Effector

The PCB design went through a few iterations as well throughout the semester as can be seen in figure 19. Originally our system was designed to work with an Arduino nano and a Pololu 1457 on the same PCB board, allowing us to control the speed of the motor with PID velocity control utilizing the motors encoders. However, due to some short circuiting issues as a result of poor use of flux and malfunctioning components (Pololu 1457), we had to redesign the PCB to move the motor controller off the PCB. We decided to use the Cytron MD10C instead of the Pololu 1457, and using some jumper wires, were able to implement the Cytron into the system and utilize it as we planned to use the Pololu. With this completed, we developed some control code for the arduino, which we based on our prior work in the sensors and motors lab. The end result was that when the PCB was powered with an external power supply, our computer was able to control the reamer velocity via a rostopic and that velocity would be maintained regardless of the torque.

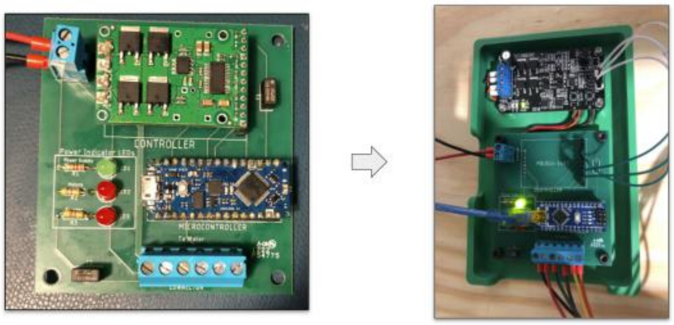


Figure 19: Evolution of the PCB

One final hardware task we would like to highlight is the set-up of the force-torque sensor. The force-torque sensor was powered using the same power supply as the reamer motor, and it was communicated with via a provided Ethernet cable. There were several methods of communicating with the force torque sensor, but the most consistent for us was logging into the sensor via a telnet session and requesting the forces and torques at a frequency of around 60 Hz. Implementing these values received from the telnet session into a ROS node allowed us to publish the resulting values to rostopics. One issue with this set-up is how the values are biased, as the values are initially biased at start-up to eliminate the forces and torques as a result of the connection to the end-effector. However, these values change, leading forces and torques being recorded when there are none acting on the system. This led to a bug in our system which caused the reamer motor to turn on early when it should have only turned on when the end-effector contacted the pelvis. To prevent this bug we would like to improve how we are handling the biasing of the data in the future.

6.4 Modeling, Analysis, and Testing

6.4.1 Motor Analysis

Some analysis we would like to highlight for our system is how we went about determining the rotational speed and torque requirements for our motor. Through some preliminary research, we determined from the paper ”A cadaver-based biomechanical model of acetabulum reaming for surgical virtual reality training simulators” by Pellicia, that the maximum torque necessary to ream 5 mm into a patient’s acetabulum is around 0.5 Nm [5]. Further research, coming primarily from the paper ”Study on cutting force of reaming porcine bone and substitute bone” by Liu, showed that in orthopedic surgery, reamers are typically driven by medical electric drills and their rotational speed is generally lower than 400 rpm [6]. As a result we determined that we needed a motor that was capable of maintaining a consistent 400 rpm at a maximum torque of 0.5 Nm.

Given these requirements, we set out to find a motor that could meet these specifications and allow us to properly ream our Sawbones. We eventually found the ServoCity Planetary Gear Motor, which had a no load speed of 612 rpm and a stall torque of 1.6 Nm. Assuming a linear relationship between the no load speed and the stall torque yielded that this motor should be able to maintain 400 rpm of rotational speed at 0.52 Nm of torque. Originally we tested this by measuring the rotational speed of the motor with a camera and the torque of the motor with a crude measuring device consisting of a custom 3D printed lever and a scale. But once we got the PCB and the force torque sensor up and running we were able to better verify the motors performance by pushing foam onto the reamer head until a torque of 0.5 Nm was measured and then checking the resulting reamer velocity according to the motor encoders. Doing so, we found that remained at a consistent 390 rpm, which could be attributed to encoder and PID control inaccuracies. As a result, we determined that we hit our specifications for the performance of the reamer system.

6.4.2 Controls Analysis

Although we currently have a PID Wrench Controller implemented for our controls subsystem, our long term goal is to use Model Predictive Control for its ability to hold constraints such as force constraints, joint limits (position, velocity, and torque constraints), and end-effector velocity constraint, while also providing optimal joint torques to reach target poses. In order to do this, we first needed to formulate the optimal control problem, which consisted of an objective function, system constraints, and system dynamics. We also analyzed literature to determine the dynamics model for drilling bone to use in our system dynamics. Most literature showed that drilling bone can be be modeled as a mass damper system, as the forces from drilling bone is proportional to velocity of drilling [7] [8]. Using our Kinova Gen 3's constraints, classic manipulator dynamics, and bone drilling dynamics, the following optimal control problem was formulated (Equation 1).

$$\begin{aligned}
\min_{s_k, u_k, k \epsilon [1,H]} \quad & \frac{1}{2}(s_H - s_d)^T Q_H (s_H - s_d) + \sum_{k=1}^{H-1} \frac{1}{2}(s_k - s_d)^T Q (s_k - s_d) + \frac{1}{2} u_k^T R u_k \\
\text{s.t.} \quad & \begin{bmatrix} \dot{q} \\ \ddot{q} \\ \dot{x} \\ \ddot{x} \\ \dot{F}_{External} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(u - \tau_{External} - C\dot{q} - G) \\ J\dot{q} \\ \dot{J}\dot{q} + J\ddot{q} \\ B_{External}(\dot{J}\dot{q} + J\ddot{q}) \end{bmatrix} \\
& u = \tau_{Applied} \leq \tau_{Max} \\
& ||F_{External}|| = ||B_{External} J\dot{q}|| \leq F_{Max} \\
& ||\dot{x}|| = ||J\dot{q}|| \leq \dot{x}_{Max} \\
& q \epsilon q_{Limits} \\
& \dot{q} \epsilon \dot{q}_{Limits}
\end{aligned} \tag{1}$$

Figure 20 shows the high-level MPC Control Block Diagram, with the MPC generating reference joint torques for the arm to follow, and joint positions, joint velocities, end-effector velocities, and forces as feedback to the MPC.

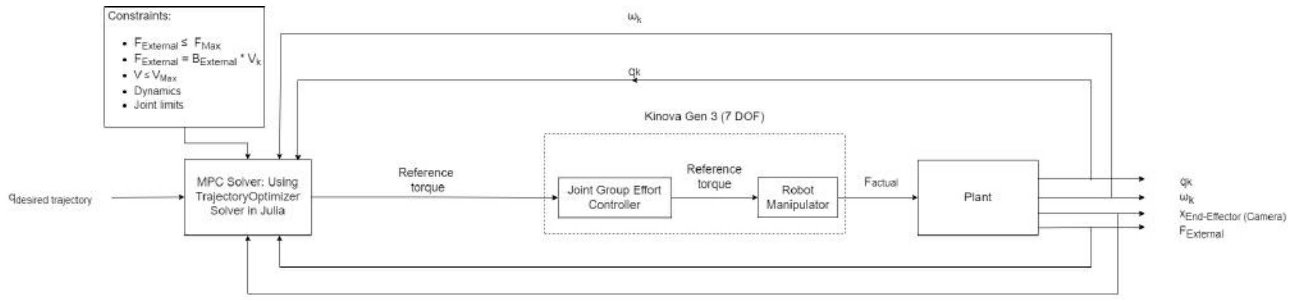

Constraints:
$F_{External} \leq F_{Max}$
$F_{External} = B_{External} * V_k$
$V \leq V_{Max}$
Dynamics
Joint limits
ω_k
q_k
Kinova Gen 3 (7 DOF)
$q_{desired\ trajectory}$
MPC Solver: Using TrajectoryOptimizer Solver in Julia
Reference torque
Joint Group Effort Controller
Reference torque
Robot Manipulator
F_{actual}
Plant
q_k
ω_k
$x_{End\text{-}Effector\ (Camera)}$
$F_{External}$

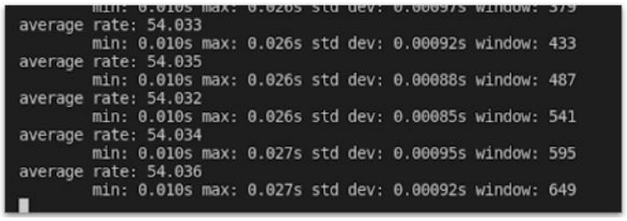
Figure 20: High-Level MPC Control Block Diagram

For the Wrench PID Controller, we did some basic behavior analysis of how Kinova's implemented wrench controller behaves. Some tests we performed were to see if the wrench controller would allow for movement in the opposite direction of its push, for which we sent a wrench command and applied opposing force, showing that the arm will not try to hold position, which was expected. Another quick test was to see if the wrench sent to the wrench controller was the wrench applied. To do this, we used the Force/Torque sensor to measure the applied forces from the wrench command. While this was not a rigorous test, we observed that the forces never exceeded the sent wrench command, and was always under. This is likely because of some modeling inaccuracies from our added end-effector. Some form of force/feedback may be implemented in future work to make the wrench controller more accurate.

6.5 Performance evaluation against the Spring Validation Demonstration (SVD)

6.5.1 Perception and Sensing

The latency requirements for perception and sensing were met with a current FPS of approximately 54-55. This is about 20 milliseconds. The latency requirements were tested based on profiling the code and using ROS frequency echo commands. The outputs are shown in Figure 21.



```
        min: 0.010s max: 0.026s std dev: 0.00097s window: 379
average rate: 54.033
        min: 0.010s max: 0.026s std dev: 0.00092s window: 433
average rate: 54.035
        min: 0.010s max: 0.026s std dev: 0.00088s window: 487
average rate: 54.032
        min: 0.010s max: 0.026s std dev: 0.00085s window: 541
average rate: 54.034
        min: 0.010s max: 0.027s std dev: 0.00095s window: 595
average rate: 54.036
        min: 0.010s max: 0.027s std dev: 0.00092s window: 649
```

Figure 21: Latency Test for Perception and Sensing

The pelvis tracker is able to detect error in the pelvis marker within position error of 2mm and orientation error of 3 degrees. The error was calculated based on the 3D printed part shown in Figure 22. Each marker slot is situated 30mm away. A triangular geometry was placed as shown in the figure, on two sides. The two poses were recorded. The test was performed 10-15 times, and the error was calculated.

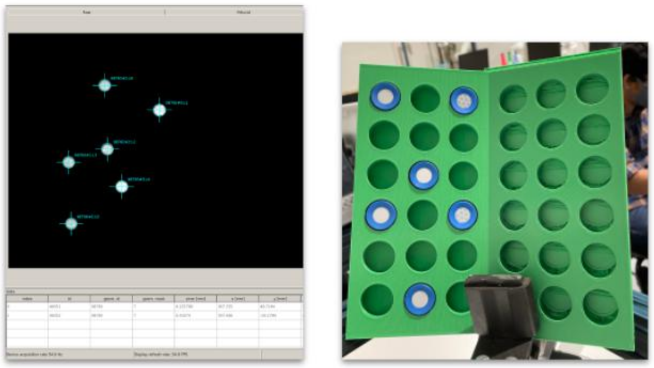


Figure 22: Pose Test for Perception and Sensing [left] Error Shown in Atracsys SDK GUI [right] 3D Printed Tested Part

6.5.2 Planning and Controls

We evaluated the performance of our planning and controls sub-systems using the RPG Trajectory Evaluation toolbox. The toolbox was used to analyze a single trajectory estimate. We provided as input the ground truth and the trajectory followed in reality as estimated by the encoders. The format in which this information had to be parsed was specific, which included the time stamp of a pose, the 3D position of the tool and the orientation in quaternion. The evaluation criteria was 'sim3', which is a similarity transformation for visual-only monocular cases. The results from the evaluation can be found in Fig. 23.

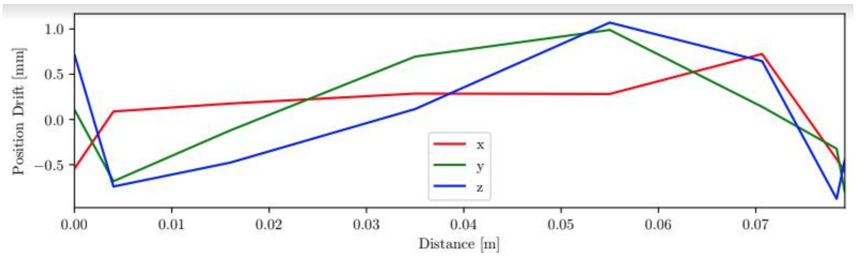


Figure 23: Position error over 10cm motion

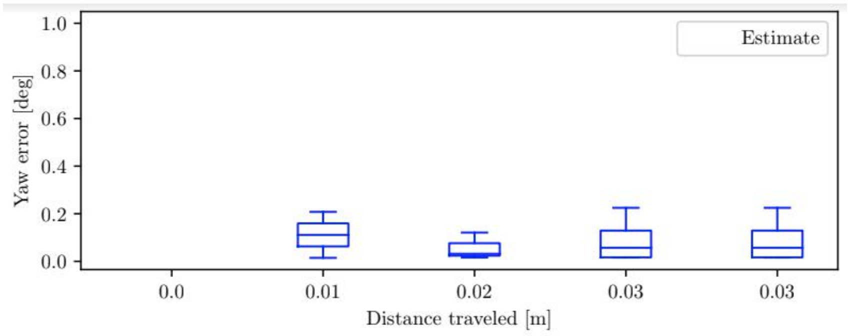


Figure 24: Yaw error over 10cm motion

From the results, it can be seen that our maximum position error (Euclidean) was **2.61mm**. Additionally, the orientation error in degrees was **0.241**. Both these errors were within our performance requirement for the Spring semester.

The planning execution time varies between **200ms to 700ms** and is highly dependent on the configuration of the robot. Based on the configuration of the robot and its proximity to a singularity, the node takes some time to plan a linear path in Cartesian space which involves computing inverse kinematics at each waypoint in the trajectory. In easier configurations the planner takes approximately 200ms to plan a linear path. The speed of the planner is also restricted by the hardware limits of the robot. As the Kinova Gen3 is a co-bot, its hardware capabilities are limited.

6.6 Strong/weak points

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

6.6.1 Perception & Sensing

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

6.6.2 Planning & Controls

- **Strengths:** The sub-systems are highly accurate and repeatable in their performance.
- **Weakness:** Planning sometimes takes more time to plan due to robot configuration. Controls tuning is time consuming and the sub-system is sensitive to the registration process.
- **Future work:** Optimize sub-system robustness and reduce latency. Reduce execution time with high accuracy.

6.6.3 Hardware

- **Strengths:** The sub-system is designed with modularity and ease-of-manufacturing in mind.
- **Weakness:** It is prone to vibrations during the reaming process.
- **Future work:** Redesign end-effector for lower vibration and improved stiffness. Rework the PCB for intended use.

7 Project Management

7.1 Work Breakdown Structure

Figure 25 depicts the high level Work Breakdown Structure required to execute and build the ARTHuR system successfully. These work packages were further broken down into lower level work packages for the schedule in Figure 28 for the sake of defining what work would need to be done every week. Generally the work is broken down into 8 categories: Hardware Setup, Simulation Setup, Perception & Sensing, Control & Actuation, Planning, Surgeon I/O, System Integration, and Management. The Work Breakdown Structure has been updated to depict progress made from Spring 2022 and anticipated tasks for Fall 2022. Most subsystem development took place in Spring 2022. These subsystems were integrated to some extent for the Spring Validation Demonstration. However, for the sake of performance optimization and integration of new features, Sub-System Integration has been marked yellow. The Surgeon I/O and Watch-Dog Modules will be the primary work effort for Fall 2022. New features developed will be integrated and Full System Integration and Testing will be performed leading up to Fall Validation Demonstration. Since ARTHuR is a performance-critical system, Performance Optimization is a continued effort. Further, some specifics of the system, for instance, the controls algorithm and reaming unit design will be revisited in Fall 2022. Finally, we anticipate Management to be a large work package as well which would involve consistent work by our Project Manager who would consistently check our progress against the work we against our schedule and analyze potential risks and mitigation actions.

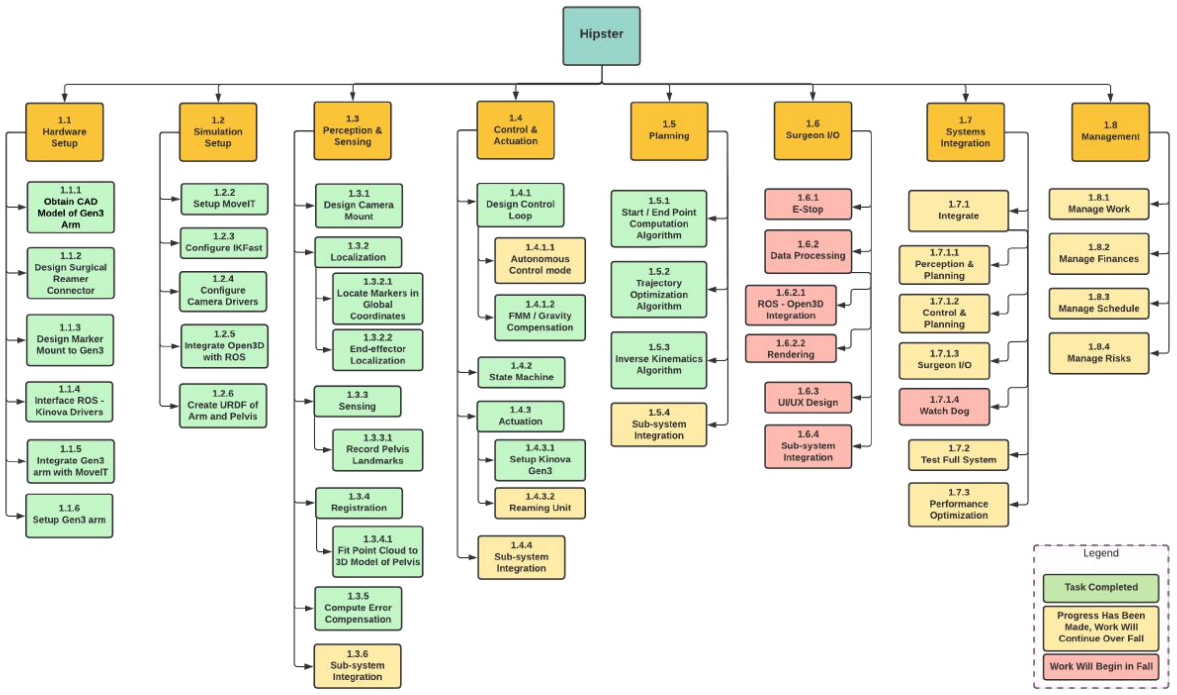

Hipster
1.1 Hardware Setup
1.1.1 Obtain CAD Model of Gen3 Arm
1.1.2 Design Surgical Reamer Connector
1.1.3 Design Marker Mount to Gen3
1.1.4 Interface ROS - Kinova Drivers
1.1.5 Integrate Gen3 arm with MoveIT
1.1.6 Setup Gen3 arm
1.2 Simulation Setup
1.2.2 Setup MoveIT
1.2.3 Configure IKFast
1.2.4 Configure Camera Drivers
1.2.5 Integrate Open3D with ROS
1.2.6 Create URDF of Arm and Pelvis
1.3 Perception & Sensing
1.3.1 Design Camera Mount
1.3.2 Localization
1.3.2.1 Locate Markers in Global Coordinates
1.3.2.2 End-effector Localization
1.3.3 Sensing
1.3.3.1 Record Pelvis Landmarks
1.3.4 Registration
1.3.4.1 Fit Point Cloud to 3D Model of Pelvis
1.3.5 Compute Error Compensation
1.3.6 Sub-system Integration
1.4 Control & Actuation
1.4.1 Design Control Loop
1.4.1.1 Autonomous Control mode
1.4.1.2 FMM / Gravity Compensation
1.4.2 State Machine
1.4.3 Actuation
1.4.3.1 Setup Kinova Gen3
1.4.3.2 Reaming Unit
1.4.4 Sub-system Integration
1.5 Planning
1.5.1 Start / End Point Computation Algorithm
1.5.2 Trajectory Optimization Algorithm
1.5.3 Inverse Kinematics Algorithm
1.5.4 Sub-system Integration
1.6 Surgeon I/O
1.6.1 E-Stop
1.6.2 Data Processing
1.6.2.1 ROS - Open3D Integration
1.6.2.2 Rendering
1.6.3 UI/UX Design
1.6.4 Sub-system Integration
1.7 Systems Integration
1.7.1 Integrate
1.7.1.1 Perception & Planning
1.7.1.2 Control & Planning
1.7.1.3 Surgeon I/O
1.7.1.4 Watch Dog
1.7.2 Test Full System
1.7.3 Performance Optimization
1.8 Management
1.8.1 Manage Work
1.8.2 Manage Finances
1.8.3 Manage Schedule
1.8.4 Manage Risks
Legend
Task Completed
Progress Has Been Made, Work Will Continue Over Fall
Work Will Begin in Fall

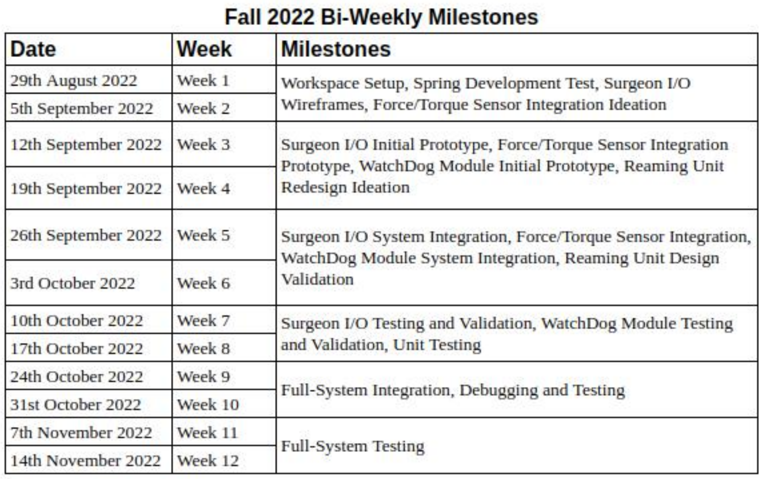
Figure 25: Work Breakdown Structure

7.2 Schedule

The bi-weekly schedule can be seen in Table 4. The major system development milestones for Fall 2022 are the Surgeon I/O and Watch Dog modules, along with Full-System Integration and Testing. Furthermore, there are also minor Sub-System level tasks. These include redesigning the reaming unit, investigating Model Predictive Control (MPC), and identifying edge cases. The Surgeon I/O will follow the new system development process. It will start with creating and iterating through design wireframes, prototyping and through the weeks, continuously testing and integrating with other subsystems. Similar to the Surgeon I/O, the WatchDog Module is also new and will require brainstorming a module design. The module will be responsible for monitoring and profiling all other sub-systems and therefore, major efforts will be geared towards seamless integration.

We aim to complete Full-System Development at least four weeks prior to Fall Validation Demonstration leaving us enough time for testing and debugging. Currently we are on schedule and will be continuing tasks as planned.

Table 4: Bi-Weekly Fall Milestones



Fall 2022 Bi-Weekly Milestones

Date	Week	Milestones
29th August 2022	Week 1	Workspace Setup, Spring Development Test, Surgeon I/O Wireframes, Force/Torque Sensor Integration Ideation
5th September 2022	Week 2	
12th September 2022	Week 3	Surgeon I/O Initial Prototype, Force/Torque Sensor Integration Prototype, WatchDog Module Initial Prototype, Reaming Unit Redesign Ideation
19th September 2022	Week 4	
26th September 2022	Week 5	Surgeon I/O System Integration, Force/Torque Sensor Integration, WatchDog Module System Integration, Reaming Unit Design Validation
3rd October 2022	Week 6	
10th October 2022	Week 7	Surgeon I/O Testing and Validation, WatchDog Module Testing and Validation, Unit Testing
17th October 2022	Week 8	
24th October 2022	Week 9	Full-System Integration, Debugging and Testing
31st October 2022	Week 10	
7th November 2022	Week 11	Full-System Testing
14th November 2022	Week 12	

7.3 Test Plan

For our Spring Validation and Demonstration (SVD), we demonstrated the development and integration of our core subsystems (with the exception of Surgeon I/O) with some relaxed performance requirements. Our efforts towards optimizing performance to achieve the latency and performance requirements will be carried out during the Fall semester.

7.3.1 Significant Testing Activities

The following is a list of crucial testing activities and their important elements:

1. **Accuracy Thresholds:** This system level test aims to improve upon the system robustness and sub-system performance. More specifically, we will again aim to hit our performance requirements of reaming to the desired end-point within a position error of **3mm** and orientation error of **3 degrees**.

2. **Planning Latency Test:** We will aim to reduce our planning latency to within 150 ms. This is an essential component to ensure that dynamic compensation works reliably. Currently, our planning sub-system using the Pilz industrial planner takes between 500-600 ms to generate a new path.

3. **Real-time Dynamic Compensation:** Given a low planning latency as described in the previous test, we will ensure that we are able to optimize our controls sub-system such that dynamic compensation can be performed in a real-time manner without having to retract the arm back as demonstrated during our SVD.

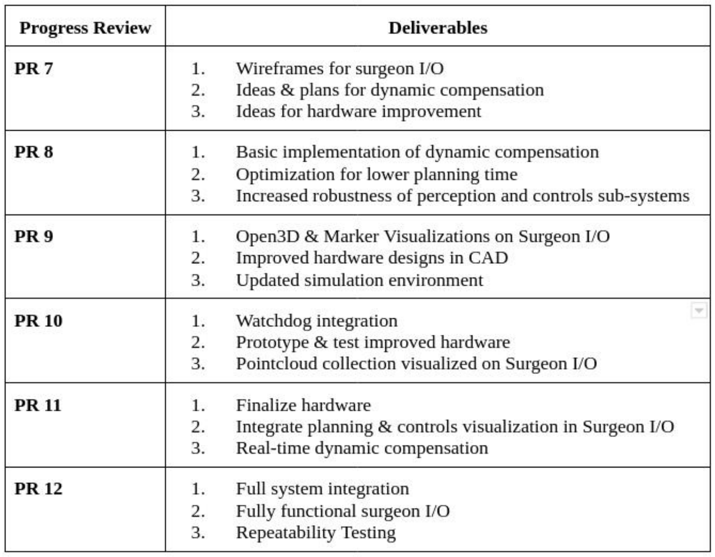
4. **Watch dog Test:** We will ensure that a watch-dog is integrated into our system such that any sub-system failures can be detected and safely acted upon within **500 ms**.

5. **Surgeon I/O Test:** We will test the capability of the surgeon I/O to integrate with the various existing sub-systems. We will further evaluate the ease of its usability through user-studies.

7.3.2 Fall Capability Milestones

Table 5 below details the capability milestones for each progress review from early September through to late November.

Table 5: Table of Key Capability Milestones for Fall Semester



Progress Review	Deliverables
PR 7	1. Wireframes for surgeon I/O 2. Ideas & plans for dynamic compensation 3. Ideas for hardware improvement
PR 8	1. Basic implementation of dynamic compensation 2. Optimization for lower planning time 3. Increased robustness of perception and controls sub-systems
PR 9	1. Open3D & Marker Visualizations on Surgeon I/O 2. Improved hardware designs in CAD 3. Updated simulation environment
PR 10	1. Watchdog integration 2. Prototype & test improved hardware 3. Pointcloud collection visualized on Surgeon I/O
PR 11	1. Finalize hardware 2. Integrate planning & controls visualization in Surgeon I/O 3. Real-time dynamic compensation
PR 12	1. Full system integration 2. Fully functional surgeon I/O 3. Repeatability Testing

7.3.3 Fall Validation Experiment

The objective of our fall validation demonstration is to demonstrate that the system is capable of autonomously localizing, planning, and executing an acetabular reaming operation as it would be performed in an operating room. We would be demonstrating this in NSH B512, utilizing all the hardware that we summarized in the full system depiction section. Our demo would consist of one team member interacting with the robot arm and work environment, one member controlling the robot arm and monitoring the Surgeon I/O, and the rest presenting and answering questions during the procedure.
Procedure:

1. Begin by setting up the work environment by clamping the Sawbone pelvis in a new position in a vise, fixing a fiducial marker screw mount on the pelvis, and placing the fiducial marker onto the end-effector of the robot arm.

2. Utilizing a probe, the pelvis will be localized using a point cloud to fit the pelvis to a known pelvis mesh, from which the endpoint of the reaming operation will be determined

3. Utilizing free motion mode, the robot arm will be placed near the center of the acetabulum.

4. The reaming operation would then be started, allowing the robot arm to localize itself with respect to the pelvis and begin generating a motion plan.

5. Once the reaming motor turns on and the arm begins to move, contacting the pelvis, the e-stop is hit to demonstrate the safety of the system.

6. The robot arm will then be reset with free motion mode and the reaming operation would then be allowed to progress freely.
7. As the robot arm begins to ream the acetabulum, the pelvis would be shifted by hand using the vise, to demonstrate the robot arm's capability of adapting to pelvic motion.
8. When the robot arm has completed the reaming operation, it will remove itself from the pelvis, and the resulting acetabulum can be analyzed.

Quantitative Performance Metrics:

- Pelvis and end-effector pose must be localized with a latency of 50 ms
- Error in pelvis pose must be calculated with a latency of 50 ms, a position error of 1 mm, and an orientation error 1.5 degrees
- Personnel should be able to move robot arm freely during free motion mode
- Once the e-stop is pressed the motor turns off and the arm stops moving within 500 ms
- Trajectory should be generated within 150 ms once the starting position is reached
- A new trajectory is planned via dynamic compensation when pelvis is moved more than 1 mm or more than 1.5 degrees, and the execution uses the new trajectory
- Using the Quantitative Trajectory Evaluator, evaluate the desired trajectory and the actual trajectory and verify the error never exceeds 1 mm in position and 1.5 degrees in orientation
- Surgeon I/O allows for control and visualization of the procedure

Graphical Representations:

The flowchart shown in Figure 26 summarizes the sequence of events during the course of FVD:

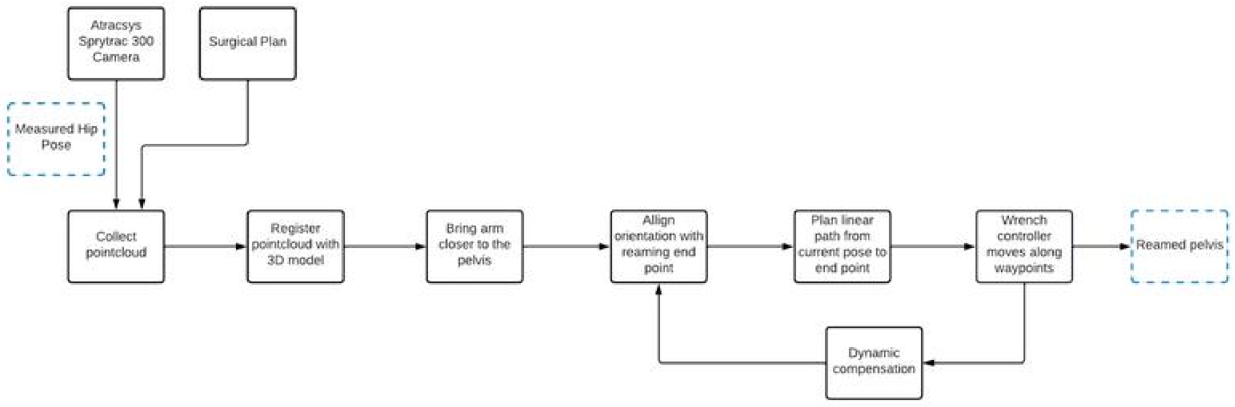

Atracsys Sprytrac 300 Camera
Surgical Plan
Measured Hip Pose
Collect pointcloud
Register pointcloud with 3D model
Bring arm closer to the pelvis
Allign orientation with reaming end point
Plan linear path from current pose to end point
Wrench controller moves along waypoints
Reamed pelvis
Dynamic compensation

Figure 26: Sequence of Events to be followed during FVD

7.4 Budget

The parts and budget list for the Spring semester is listed in Table 6 with a total budget of $5000. Although, all the necessary equipment to complete the project was provided to us by our sponsors, we are in need to buy model bones for testing and other accessories to complement the hardware, which comprised of the majority of our budget till now. For the fall semester, we would have to buy more bone models for testing and demonstration purposes. We have spent a total of **11.50%** of our budget until now, retaining 88.50%. We are trying to retain a majority of the project funds for risk mitigation strategies involving hardware failure such as camera failure and robot arm malfunctioning.

Table 6: Parts List and Budget

Item	Unit Cost($)	Quantity	Total Cost
Hemi Pelvis 48mm - Solid foam	27.10	5	135.5
Hemi Pelvis 56mm with vise attachment - Solid foam	27.10	5	135.5
Monitor Desk Mount Vesa	42.40	1	42.4
Planetary Gear Motor	59.99	1	59.99
Motor coupling	7.33	3	21.97
Motor mounting hub	7.95	1	7.95
PLA material	23.99	1	23.99
Ethernet splitter	17.99	1	17.99
Power strip	14.56	1	14.56
Cord sleeve	15.98	1	15.98
Surgeon gown	27.99	1	27.99
HDMI Cable	7.30	1	7.30
Conference Mic	32.99	1	32.99
Presentation Pointer	29.99	1	29.99
Power strip waterproof sleeve	19.99	1	19.99
Late july chips	39.50	1	39.50
Total			**$574.22**

7.5 Risk Management

There are many technical, schedule, and programmatic risks that could adversely effect the team's capability to complete the project successfully, all of which are summarized in Figure 34. From the risks listed in the table, we realized the risk of the robot arm not arriving on time. This risk could have caused significant delays and major re-scoping of the problem statement if we had not done our due diligence as part of the Systems Engineering course. Below are the effects and the mitigation strategies applied:
Risks realized:

1. The Link-6 arm from Kinova arrived on Feb 5th, as opposed to the expected date of Jan 20th.

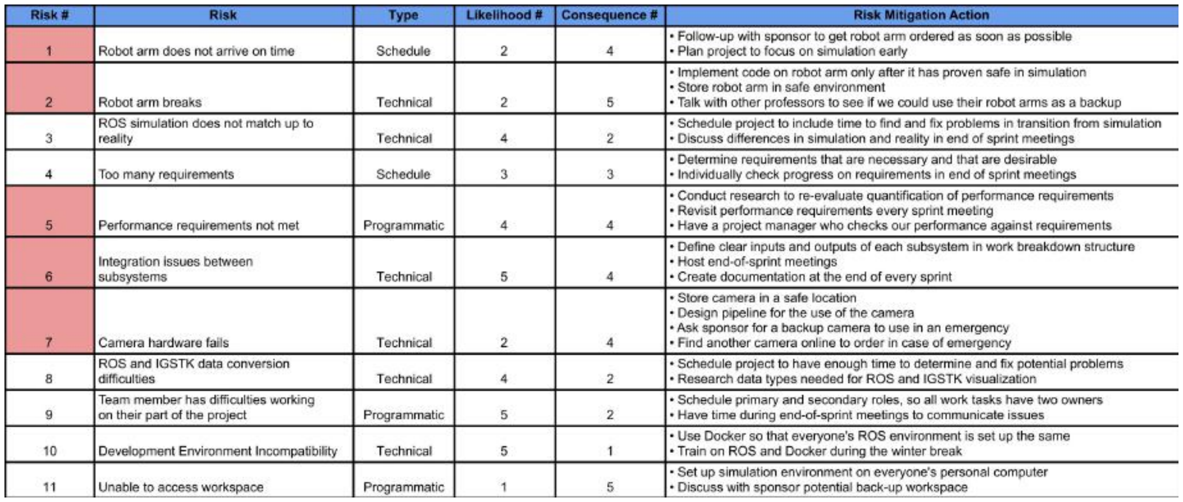
2. The Link-6 had deficiencies in its software capabilities and could not be integrated with ROS, which was a crucial factor in our system performance.

3. The Kinova Gen3 arm arrived two weeks after us returning the Link-6 to our sponsors, leading to a total delay of 4 weeks in receiving the arm.

Mitigation steps:

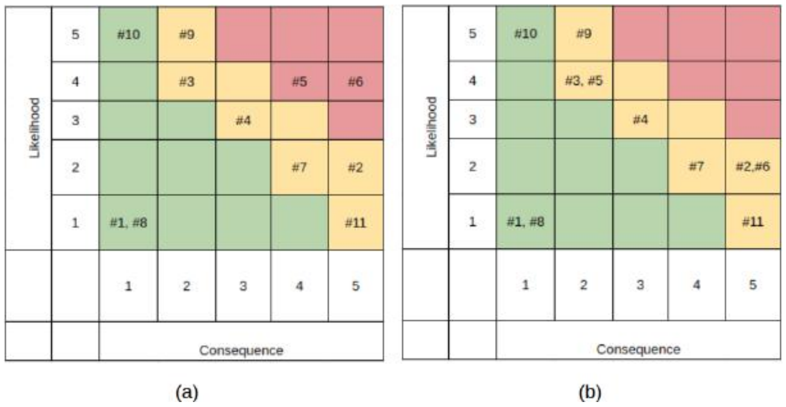
1. We started developing our sub-systems in simulation, which helped us to continue working on the project and also helped us evaluate the performance of each system before integrating it in reality.

2. Frequent and clear communication with our sponsors proved crucial in getting a replacement arm in Gen3, without which meeting our requirements would have been tough.



Risk #	Risk	Type	Likelihood #	Consequence #	Risk Mitigation Action
1	Robot arm does not arrive on time	Schedule	2	4	• Follow-up with sponsor to get robot arm ordered as soon as possible • Plan project to focus on simulation early
2	Robot arm breaks	Technical	2	5	• Implement code on robot arm only after it has proven safe in simulation • Store robot arm in safe environment • Talk with other professors to see if we could use their robot arms as a backup
3	ROS simulation does not match up to reality	Technical	4	2	• Schedule project to include time to find and fix problems in transition from simulation • Discuss differences in simulation and reality in end of sprint meetings
4	Too many requirements	Schedule	3	3	• Determine requirements that are necessary and that are desirable • Individually check progress on requirements in end of sprint meetings
5	Performance requirements not met	Programmatic	4	4	• Conduct research to re-evaluate quantification of performance requirements • Revisit performance requirements every sprint meeting • Have a project manager who checks our performance against requirements
6	Integration issues between subsystems	Technical	5	4	• Define clear inputs and outputs of each subsystem in work breakdown structure • Host end-of-sprint meetings • Create documentation at the end of every sprint
7	Camera hardware fails	Technical	2	4	• Store camera in a safe location • Design pipeline for the use of the camera • Ask sponsor for a backup camera to use in an emergency • Find another camera online to order in case of emergency
8	ROS and IGSTK data conversion difficulties	Technical	4	2	• Schedule project to have enough time to determine and fix potential problems • Research data types needed for ROS and IGSTK visualization
9	Team member has difficulties working on their part of the project	Programmatic	5	2	• Schedule primary and secondary roles, so all work tasks have two owners • Have time during end-of-sprint meetings to communicate issues
10	Development Environment Incompatibility	Technical	5	1	• Use Docker so that everyone's ROS environment is set up the same • Train on ROS and Docker during the winter break
11	Unable to access workspace	Programmatic	1	5	• Set up simulation environment on everyone's personal computer • Discuss with sponsor potential back-up workspace

Figure 27: Risk Table for Spring Semester



(a)

Likelihood	1	2	3	4	5
5	#10	#9			
4		#3		#5	#6
3			#4		
2				#7	#2
1	#1, #8				#11

(b)

Likelihood	1	2	3	4	5
5	#10	#9			
4		#3, #5			
3			#4		
2				#7	#2,#6
1	#1, #8				#11

Figure 28: (a) Risks identified (b) Risks after mitigation

8 Conclusions: Lessons Learned

1. **Iterate and fail quickly:** We found ourselves trying out methods and techniques that none of the team-members had prior experience with. The Model Predictive Controller in the

controls sub-system consumed a significant chunk of time. We should have pivoted to the new PID controller sooner to allow for sufficient time for testing. This was also the case with the hardware sub-system when we had designed an effector using the reamer handle our sponsors had provided. However, we realized that it was far too long and heavy for our robot. This process could have been iterated much more quickly.

2. **Better use of Project Management Software:** While we were consistent in our use of Jira to track key milestones and plan the tasks for the next two-week sprint, we did not update our progress as consistently as we would have liked. We also didn't log the amount of time it took to accomplish each of the listed tasks consistently. These are aspects that we would like to change moving forward into the fall semester.

3. **Better Knowledge Transfer:** As a team, we worked autonomously for the most part on our assigned sub-systems in smaller groups of two people. As a result, not all team members were fully aware of the specifics being developed in other sub-systems. Moving into the fall semester, we will need to work more collaboratively since most sub-systems have been integrated. We plan to have more knowledge-sharing sessions as well provide opportunities to team-members on work on other sub-systems.

4. **Rigorous testing framework:** Having developed all the major sub-systems, we did not test edge-cases as rigorously as we would have liked due to the lack of time. Next semester, we will setup unit-testing frameworks for every sub-system to ensure that every sub-system is tested rigorously for edge-cases before a demonstration.

9 Conclusions: Key Fall Activities

1. **Increasing System Robustness:** Having completed a preliminary implementation of most major sub-systems as well the integration between them, we need to address edge cases and increase system robustness through rigorous testing.

2. **Surgeon I/O:** This is the only major sub-system which hasn't been worked upon yet. We will begin the fall semester by working on the surgeon I/O that integrates our various sub-systems coherently.

3. **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 exploring moving to MuJoCo for better controls simulation.

4. **Experimenting with other control techniques:** While we have made significant progress in our MPC controller, we haven't been able to resolve some aspects of the controller that will allow real-time implementation on the robot. We would like to fix these issues to have a more sophisticated controls implementation.

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] Stryker, “Mako tha surgical technique.” Available at `https://www.strykermeded.com/media/2042/mako-tha-surgical-technique.pdf`.

[5] C.-E. H. M. K. P. K. S. K. S. S. C. R. R. W. M. W. G. Z. D. Z. . N. H. Luigi Pelliccia, Mario Lorenz, “A cadaver-based biomechanical model of acetabulum reaming for surgical virtual reality training simulators,” *Sci Rep*, no. 10, 2020.

[6] Z. Liu, J. Sui, B. Chen, Z. Yuan, C. Du, C. Wang, and H. Chen, “Study on cutting force of reaming porcine bone and substitute bone,” *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 236, no. 1, pp. 94–102, 2022.

[7] T. Boiadjiev, G. Boiadjiev, K. Delchev, I. Chavdarov, and R. Kastelov, “Feed rate control in robotic bone drilling process,” *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 235, no. 3, pp. 273–280, 2021. PMID: 33231113.

[8] W. Wang, Y. Shi, N. Yang, and X. Yuan, “Experimental analysis of drilling process in cortical bone,” *Medical Engineering Physics*, vol. 36, no. 2, pp. 261–266, 2014.

10 Appendix

Full Size Use-case Graphic

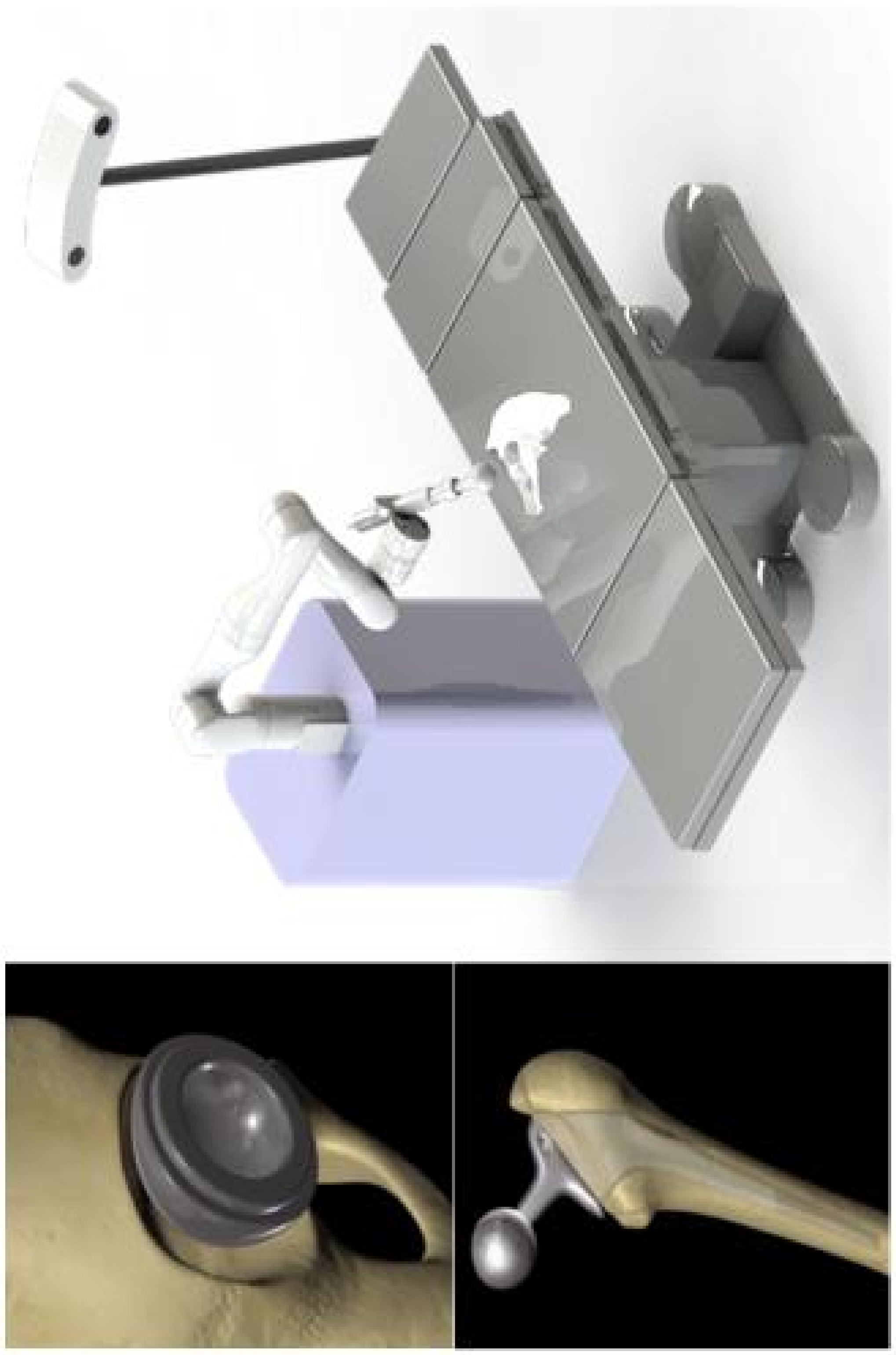


Figure 29: Full Size Use-case Graphic

Full Size Functional Architecture

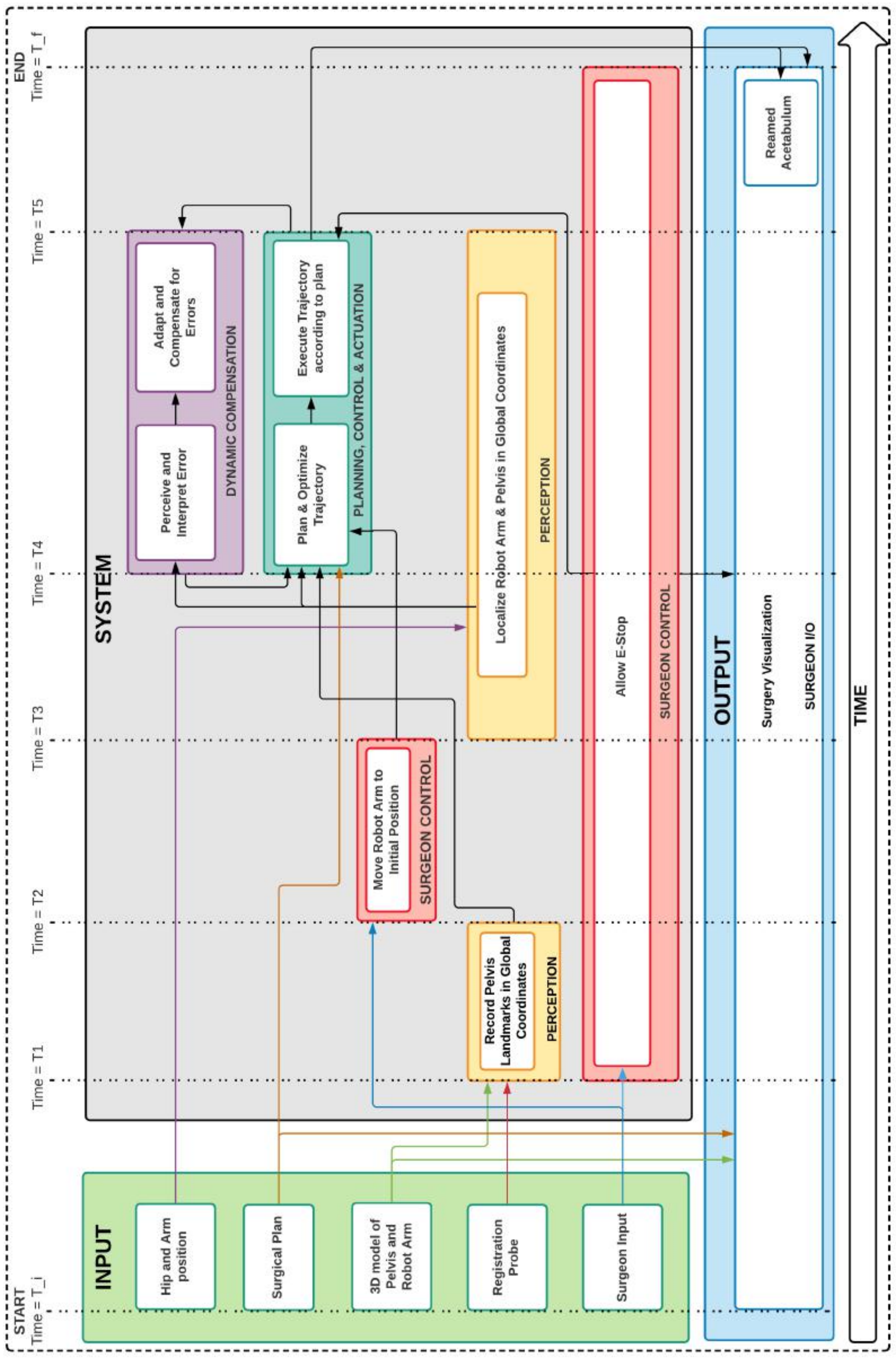

START
Time = T_i
Time = T1
Time = T2
Time = T3
Time = T4
Time = T5
END
Time = T_f
INPUT
Hip and Arm position
Surgical Plan
3D model of Pelvis and Robot Arm
Registration Probe
Surgeon Input
SYSTEM
Record Pelvis Landmarks in Global Coordinates
PERCEPTION
Move Robot Arm to Initial Position
SURGEON CONTROL
Localize Robot Arm & Pelvis in Global Coordinates
PERCEPTION
Perceive and Interpret Error
Adapt and Compensate for Errors
DYNAMIC COMPENSATION
Plan & Optimize Trajectory
Execute Trajectory according to plan
PLANNING, CONTROL & ACTUATION
Allow E-Stop
SURGEON CONTROL
OUTPUT
Surgery Visualization
SURGEON I/O
Reamed Acetabulum
TIME

Figure 30: Full Size Functional Architecture : Revised

Full Size Cyberphysical Architecture

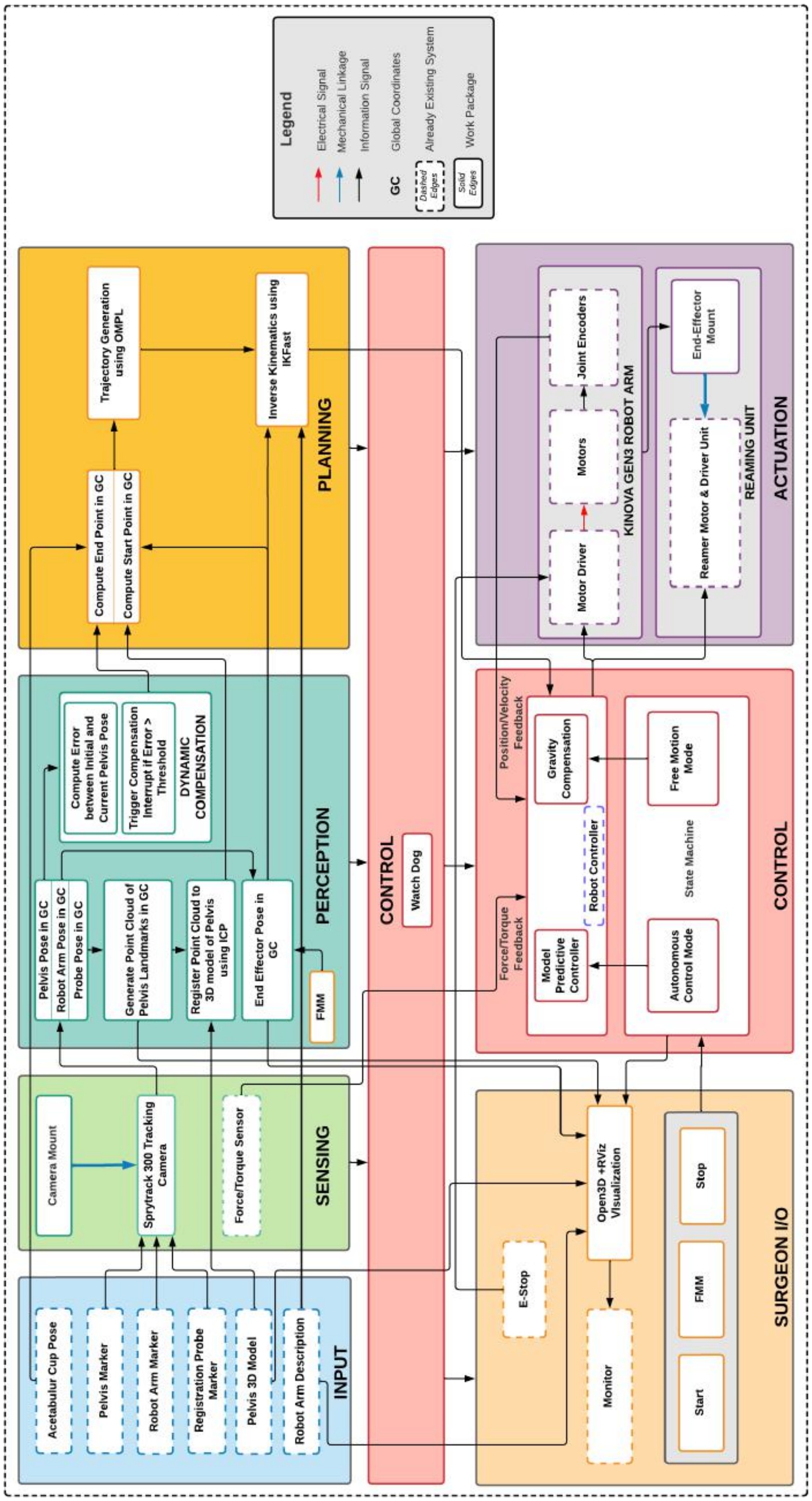


Figure 31: Full Size Cyberphysical Architecture

Full Size Work Breakdown Structure

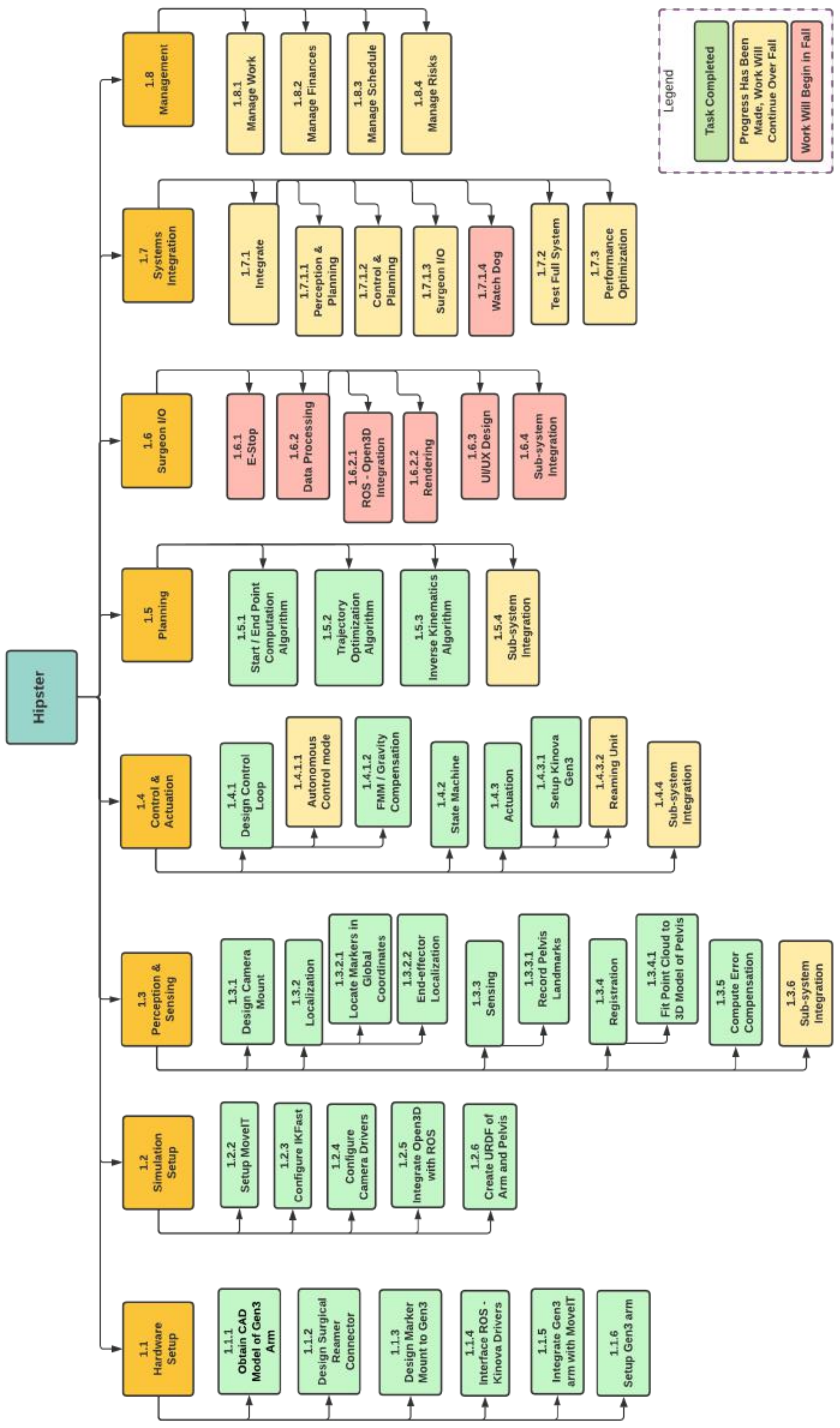

Hipster
1.1 Hardware Setup
1.1.1 Obtain CAD Model of Gen3 Arm
1.1.2 Design Surgical Reamer Connector
1.1.3 Design Marker Mount to Gen3
1.1.4 Interface ROS - Kinova Drivers
1.1.5 Integrate Gen3 arm with MoveIT
1.1.6 Setup Gen3 arm
1.2 Simulation Setup
1.2.2 Setup MoveIT
1.2.3 Configure IKFast
1.2.4 Configure Camera Drivers
1.2.5 Integrate Open3D with ROS
1.2.6 Create URDF of Arm and Pelvis
1.3 Perception & Sensing
1.3.1 Design Camera Mount
1.3.2 Localization
1.3.2.1 Locate Markers in Global Coordinates
1.3.2.2 End-effector Localization
1.3.3 Sensing
1.3.3.1 Record Pelvis Landmarks
1.3.4 Registration
1.3.4.1 Fit Point Cloud to 3D Model of Pelvis
1.3.5 Compute Error Compensation
1.3.6 Sub-system Integration
1.4 Control & Actuation
1.4.1 Design Control Loop
1.4.1.1 Autonomous Control mode
1.4.1.2 FMM / Gravity Compensation
1.4.2 State Machine
1.4.3 Actuation
1.4.3.1 Setup Kinova Gen3
1.4.3.2 Reaming Unit
1.4.4 Sub-system Integration
1.5 Planning
1.5.1 Start / End Point Computation Algorithm
1.5.2 Trajectory Optimization Algorithm
1.5.3 Inverse Kinematics Algorithm
1.5.4 Sub-system Integration
1.6 Surgeon I/O
1.6.1 E-Stop
1.6.2 Data Processing
1.6.2.1 ROS - Open3D Integration
1.6.2.2 Rendering
1.6.3 UI/UX Design
1.6.4 Sub-system Integration
1.7 Systems Integration
1.7.1 Integrate
1.7.1.1 Perception & Planning
1.7.1.2 Control & Planning
1.7.1.3 Surgeon I/O
1.7.1.4 Watch Dog
1.7.2 Test Full System
1.7.3 Performance Optimization
1.8 Management
1.8.1 Manage Work
1.8.2 Manage Finances
1.8.3 Manage Schedule
1.8.4 Manage Risks
Legend
Task Completed
Progress Has Been Made, Work Will Continue Over Fall
Work Will Begin in Fall

Figure 32: Full Size Work Breakdown Structure

Full Size Project Fall Schedule

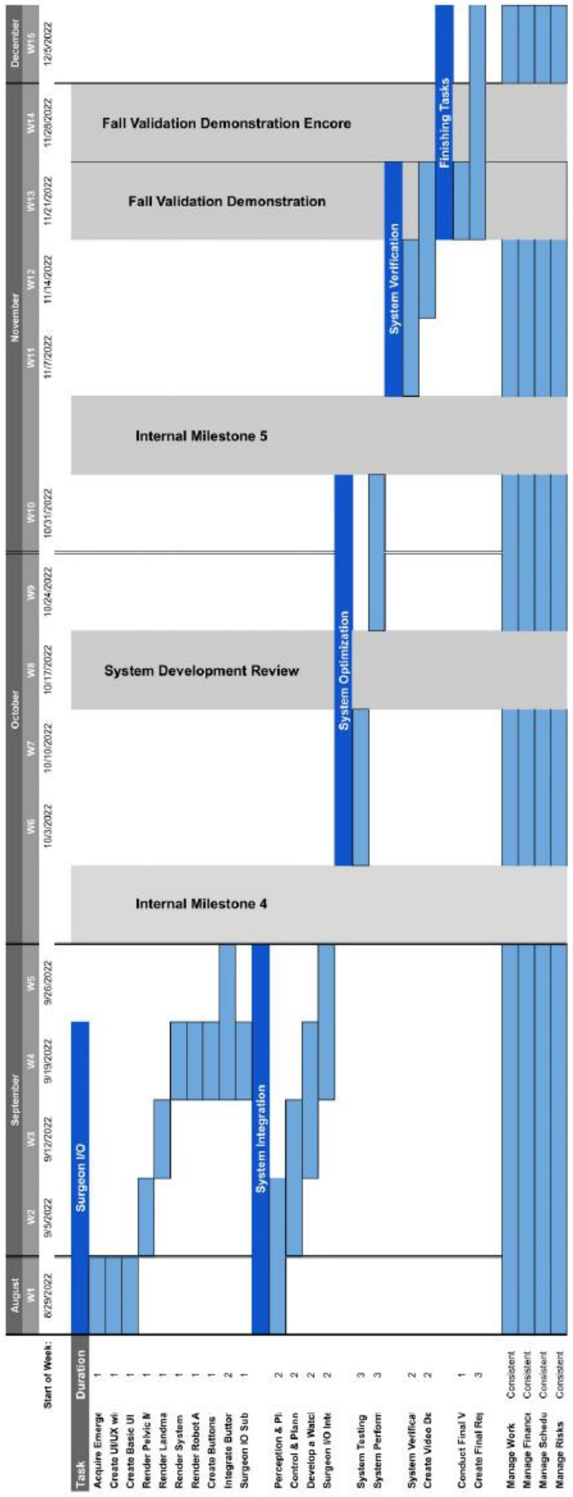


Figure 33: Full Size Project Schedule

Full Size Project Risks Summary

Risk #	Risk	Type	Likelihood #	Consequence #	Risk Mitigation Action
1	Robot arm does not arrive on time	Schedule	2	4	• Follow-up with sponsor to get robot arm ordered as soon as possible • Plan project to focus on simulation early
2	Robot arm breaks	Technical	2	5	• Implement code on robot arm only after it has proven safe in simulation • Store robot arm in safe environment • Talk with other professors to see if we could use their robot arms as a backup
3	ROS simulation does not match up to reality	Technical	4	2	• Schedule project to include time to find and fix problems in transition from simulation • Discuss differences in simulation and reality in end of sprint meetings
4	Too many requirements	Schedule	3	3	• Determine requirements that are necessary and that are desirable • Individually check progress on requirements in end of sprint meetings
5	Performance requirements not met	Programmatic	4	4	• Conduct research to re-evaluate quantification of performance requirements • Revisit performance requirements every sprint meeting • Have a project manager who checks our performance against requirements
6	Integration issues between subsystems	Technical	5	4	• Define clear inputs and outputs of each subsystem in work breakdown structure • Host end-of-sprint meetings • Create documentation at the end of every sprint
7	Camera hardware fails	Technical	2	4	• Store camera in a safe location • Design pipeline for the use of the camera • Ask sponsor for a backup camera to use in an emergency • Find another camera online to order in case of emergency
8	ROS and IGSTK data conversion difficulties	Technical	4	2	• Schedule project to have enough time to determine and fix potential problems • Research data types needed for ROS and IGSTK visualization
9	Team member has difficulties working on their part of the project	Programmatic	5	2	• Schedule primary and secondary roles, so all work tasks have two owners • Have time during end-of-sprint meetings to communicate issues
10	Development Environment Incompatibility	Technical	5	1	• Use Docker so that everyone's ROS environment is set up the same • Train on ROS and Docker during the winter break
11	Unable to access workspace	Programmatic	1	5	• Set up simulation environment on everyone's personal computer • Discuss with sponsor potential back-up workspace

Figure 34: Full Size Project Risks Summary