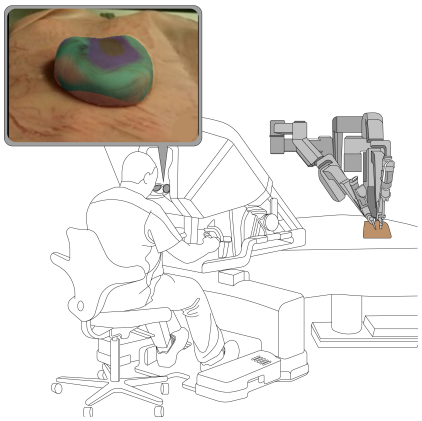
Carnegie Mellon University

MRSD Project Course II

December 16, 2020



Final Report

Augmented Reality for Minimally Invasive Tumor Resection:
The Chopsticks Robotic Surgical System

Team A
Anjali Pemmaraju
Arti Anantharaman
Alex Wu
Beilei Zhang
Chang Shi

Sponsor
Howie Choset

Mentors
Dimi Apostolopoulos
John Dolan
Nico Zevallos

1 Abstract

In the US, there are approximately 42,000 new cases of liver cancer annually. Current procedures to remove liver tumors are inefficient in how they require surgeons to manually palpate the organ to find tumors, leading to healthy tissue resection and a high surgeon cognitive load. This procedure also requires a large incision to be made so the surgeon can palpate the organ. This combined with healthy tissue resection leads to worse patient outcomes. There is a clear need to alleviate some of the burdens of the surgeon and to improve patient outcomes. This need can be fulfilled by an autonomous minimally-invasive approach that can take some of the responsibility of the surgeon and perform the task more accurately.

We are developing the Chopsticks Surgical System, an add-on to the da Vinci Surgical System, that can autonomously localize embedded liver tumors. The final iteration of this project will be able to generate a model of a silicone liver phantom in real-time and use this model to inform a palpation procedure that will find the embedded tumors. This report elucidates the approach we have taken to creating this product, from a technical and systems engineering perspective.

Contents

2 Project Description

In typical tumor resection procedures, a surgeon has to make a large incision to gain access to the diseased organ and will manually palpate the organ to locate embedded tumors. Regions of abnormally high stiffness in the organ indicate the presence of tumors. This method is not precise and leads the surgeon to resect to healthy tissue to account for a margin of error and a high surgeon cognitive load. In addition to this operative method prolonging the hospitalization time of the patient, the supplementary preoperative images such as CT scans for tumor localization make the surgical procedure a costly affair. CT scans are expensive in the United States, with an average cost of $3,275; moreover, the patient must be hospitalized for longer, which costs more and contributes to a higher risk of infection because of the large incision needed for the non-laparoscopic surgery.

The Chopsticks Robotic Surgical System can overcome these limitations of conventional tumor localization. After attaching it to a da Vinci Surgical System, the robot circumvents the need for preoperative images of the organ by constructing a three-dimensional intraoperative organ model in real-time while accounting for the natural movements of the organ. Furthermore, the robot can palpate the organ to search for tumors and overlay the location and shape of the found tumors onto the visual feed for the surgeon to see in real-time, fully automating the process of localizing tumors.

3 Use Case

Dr. A is setting up a tumor resection for his patient, Mr. B, who has been diagnosed with liver cancer. To remove the cancerous cells, Dr. A needs to reliably localize tumors in the liver and he plans to use the Chopsticks Surgical system to do so, depicted below in Figure 1.

Dr. A prepares Mr. B for his surgery with the Chopsticks Surgical robotic system. He positions the tooltips of Chopsticks close to the organ of interest and positions the stereo camera to look at the entire organ scene by manipulating the da Vinci Surgical System arms manually. He then activates Chopsticks, which gets to work immediately. One arm of Chopsticks performs a laser scan of organs in the frame and generates 2D line scans; this information is augmented with the view of the organ from the stereo camera to segment out the organ of interest. In our case, we assume this organ to be the liver. The combination of these two sensors' data yields an accurate representation of the intraoperative model of the liver, regardless of organ motion within the body. The second arm of Chopsticks, which has a force sensor mounted on its tip, then autonomously palpates the specified portion of the liver to check for the presence of embedded tumors. Higher force readings indicate stiffer tissue and the presence of tumors. The obtained force readings are converted to a heat-map representing relative stiffness in the region specified by Dr. A. This is overlaid onto the live video feed from the left and right cameras.

Dr. A, who can see the exact shape and location of tumors in the liver on his camera feed, now has sufficient information to begin the tumor resection procedure.

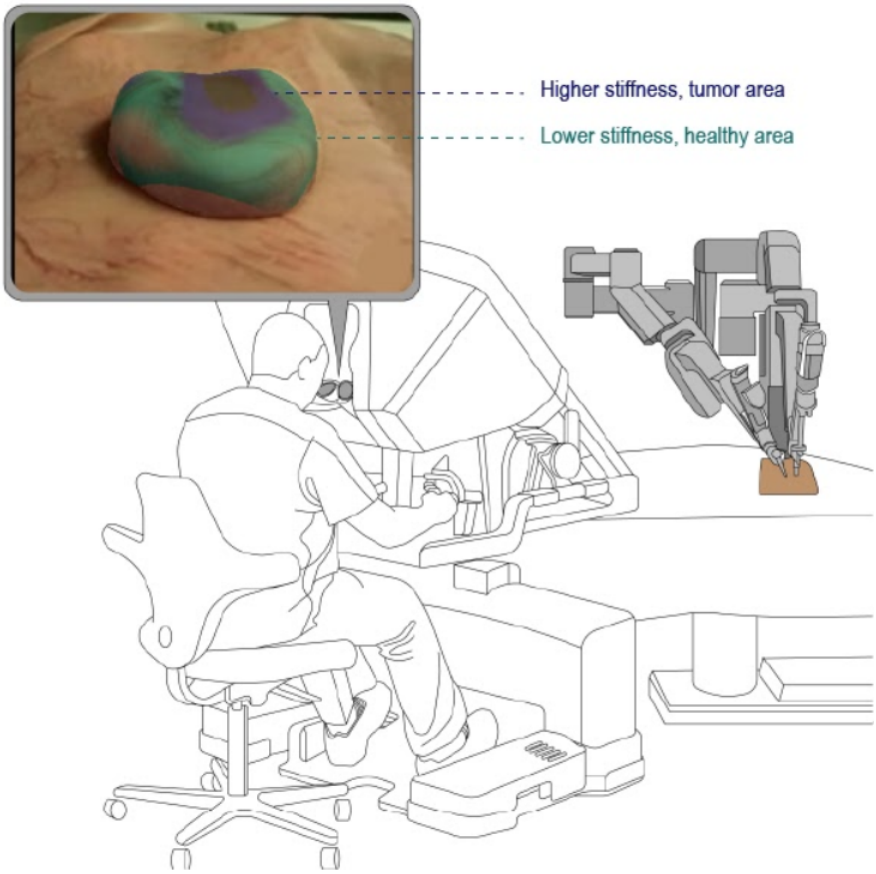


Figure 1: The Chopsticks Surgical System

4 System-Level Requirements

We derived requirements for the project from the needs analysis, by reading seminal papers [1], [2], and [3] in related fields, and through detailed discussions with our sponsor and staff at the Biorobotics Lab in Carnegie Mellon University. We have made six changes to the system-level requirements since the Conceptual Design Review in Fall 2019, most recently at the beginning of Fall 2020.

1. We discovered that it was more useful to register the laser-robot frames as opposed to the camera-laser frames. We now quantify the laser-robot registration in terms of a reprojection error, and so "M.P.2: Register camera-laser coordinate frames with a 10 mm maximum RMSE" was changed to"M.P.2: Register laser-robot coordinate frames with a 10-pixel-unit maximum reprojection error".

2. We perform liver segmentation in the 2D image space, making the previous accuracy metric in terms of voxel units unsuitable. Hence, "M.P.4: Segment the liver, including at locations within 5 mm organ occlusions, at 95% voxel accuracy" was changed to "M.P.4: Segment the liver, including near organ occlusions, at 95% IOU".

3. Our project was the first to use the Blaser sensor in conjunction with the dVRK, so we felt it would be prudent to allow for an increased range of error when using this sensor. Hence, "M.P.5: Generate a point cloud of the phantom liver within

5 mm RMSE when compared with known geometry" was revised to "M.P.5: Generate a point cloud of the phantom liver within 10 mm RMSE when compared with known geometry".

4. Since the point cloud stitching code generates a point cloud of the liver in real-time, we removed the requirement that mandated that the point cloud of the liver be generated in under 1 minute.

5. During the system conceptualization process in Fall 2019, we were unsure of how to quantify the error of liver motion - as a frequency or as an offset error. We realized it was reasonable to quantify it as a frequency error since the Fast Fourier Transform that we apply to the liver motion outputs its frequency spectrum. Therefore, "M.P.6: Estimate the motion of the phantom liver within 0.5 Hz and a 5 mm maximum RMSE" was contracted to "M.P.6: Estimate the motion of the phantom liver within 0.5 Hz".

6. We removed our previous requirement, " M.P.8: Palpate the organ with a force of <10 N so that the organ resumes its original shape" because we did not simulate a force sensor.

The revised system-level requirements of our project are detailed below.

4.1 Performance Requirements

4.1.1 Mandatory Performance Requirements

Mandatory performance requirements are detailed in Table 1.

Table 1: Mandatory Performance Requirements - Subsystem Level

Subsystem	Requirements
Data Processing & Organ Modeling	M.P.1 - Register camera-robot coordinate frames with a 10 mm maximum root-mean-square-error (RMSE) M.P.2 - Register laser-robot coordinate frames with a 10-pixel-unit maximum reprojection error M.P.3 - Scan the surface of the phantom liver in under 5 minutes M.P.4 - Segment the liver, including near organ occlusions, at 95% IOU M.P.5 - Generate a point cloud of the phantom liver within 10 mm RMSE when compared with known geometry M.P.6 - Estimate the motion of the phantom liver within 0.5 Hz
Searching & Palpating	M.P.7 - Palpate the phantom liver in under 10 minutes M.P.8 - Complete the entire surgical procedure in under 20 minutes M.P.9 - Detect >90% of cancerous tissue M.P.10 - Misclassify <10% of healthy tissue as cancerous tissue

4.1.2 Desirable Performance Requirements

After achieving the requirements listed above, we will pursue the following desirable requirements to enhance the system's capabilities. In particular, we have detailed tighter constraints on permissible errors to make the system more robust. The revised metrics are reflected in the Desirable Performance Requirements shown in Table 2.

Table 2: Desirable Performance Requirements - Subsystem Level

Subsystem	Requirements
Data Processing & Organ Modeling	D.P.1 - Register camera-robot coordinate frames with a 2 mm maximum RMSE D.P.2 - Register laser-robot coordinate frames with a 2-pixel-unit maximum reprojection error D.P.3 - Scan the surface of the phantom liver in under 1 minute D.P.4 - Generate a point cloud of phantom liver within 2 mm RMSE when compared with known geometry D.P.5 - Estimate the motion of the phantom liver within 0.2 Hz
Searching & Palpating	D.P.6 - Palpate the phantom liver in under 5 minutes D.P.7 - Complete the entire surgical procedure in under 15 minutes D.P.8 - Detect >95% of cancerous tissue D.P.9 - Misclassify <5% of healthy tissue as cancerous tissue

4.2 Non-Functional Requirements

4.2.1 Mandatory Non-Functional Requirements

These requirements apply to the whole system and increase its utility.

M.N.1 - Comply with Food and Drug Administration (FDA) regulations
M.N.2 - Reduce the cognitive overload of surgeon
M.N.3 - Not occlude the phantom liver's visibility to the surgeon
M.N.4 - Exit autonomous mode immediately upon prompt
M.N.5 - Yield itself to regular safety checks
M.N.6 - Cost less than $5000

4.2.2 Desirable Non-functional Requirements

After achieving the requirements listed above, we will attempt to enhance the system's safety capabilities by pursuing the following desirable requirements.

D.N.1 - Display an error message if the phantom organ is placed out of the reach of the robot
D.N.2 - Be able to pause its operation upon request and pick up from where it left off when its operation is resumed
D.N.3 - Palpate a given point again if the first force reading was noisy

5 Functional Architecture

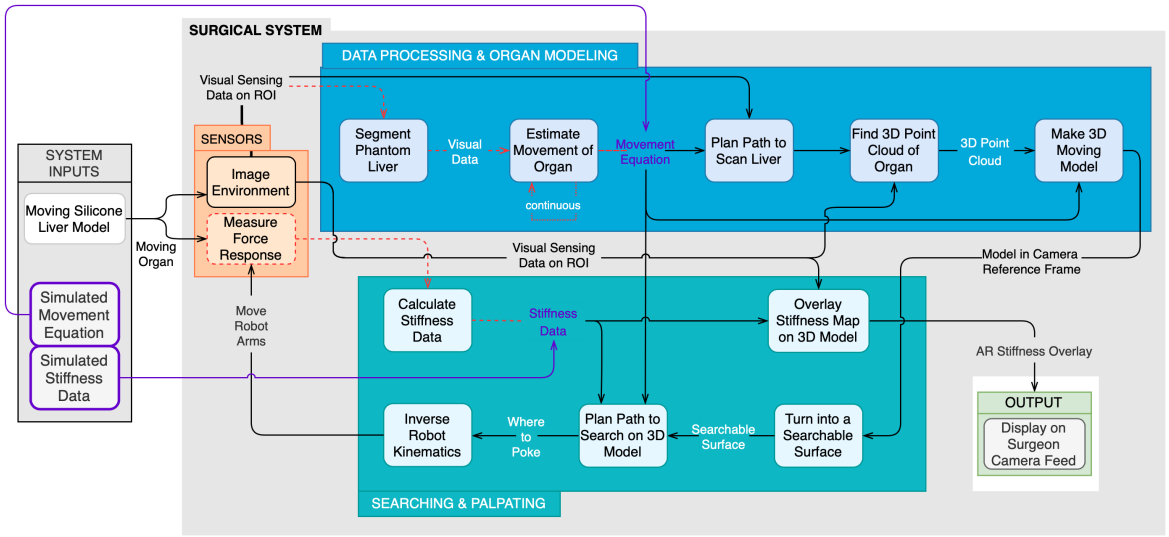


Figure 2: Chopsticks Surgical System Functional Architecture

The functional architecture in Figure 2 demonstrates the function of the Chopsticks Surgical system. Knowing that our system would be run in simulation, the purple lines represent simulated inputs that we would not otherwise be receiving in the real system. This system is installed on a da Vinci Research Kit (dVRK). It has three main subsystems and is a visual representation of the flow of data throughout our system. The first subsystem comprises the sensors. These sensors are used to determine the visual and physical properties of the phantom organ scene. More specifically, the sensors will measure the force response of and image the phantom organs. The sensed signals will be passed to the data processing & organ modeling subsystem and the searching & palpating subsystem. The final subsystem we have developed is the movement simulation platform subsystem, but it is not part of the Chopsticks Surgical System and so is not represented here.

The data processing & organ modeling subsystem is broken into two steps. The first step is to align the data from the multiple visual imaging sensors into a consistent frame, and the second is to use the visual data to determine the shape and movement of the organ. Each block within the subsystem represents a specific algorithm. The sensing information is fed into the first block, which computes transformations between the frames of each sensor, and then transformed in the next block. These two blocks represent the data processing portion of this subsystem.

Afterwards, the processed visual data are used to segment the phantom liver from the surrounding organs. The segmented data are used to estimate the movement of the organ. This equation and the visual data are used to plan a path to move the laser scanner to determine the point cloud. The final block combines the movement equation and the point cloud to create a moving 3D model of the organ. This model is the final output of the data processing & organ modeling subsystem. It gets passed on to the searching & palpating subsystem.

Using the point cloud that represents the moving organ, the first block in the searching & palpating subsystem creates a searchable surface for use in the rest of the subsystem. The search, robot kinematics, force reading transformation, and stiffness data calculation blocks form a loop. This cycle reflects how the system transforms each force reading into stiffness data and uses this to reason about where to poke next. After an adequate amount of stiffness data have been accumulated, the stiffness data are overlaid onto the point cloud outputted by the data processing & organ modeling subsystem. The stiffness map and the point cloud are the final outputs of the entire subsystem. These are rendered on the surgeon's camera feed.

6 Trade Studies

6.1 System-Level Trade Study

We performed a system-level trade study in order to determine the best system for finding embedded tumors. The main options we considered were a: Invasive surgical procedure (the doctor palpates the organs with their hands), b: Semi-autonomous robotic system, and c: Fully-autonomous robotic system. Table 3 shows the results of the trade study. The most important criteria that we based our trade study include capturing the rhythm of the moving organ almost perfectly (M.P.6), estimating the shape and size of embedded tumors (M.P.9, M.P.10), providing a visual of the stiffness map to the surgeon, and the surgeon's cognitive overload (M.N.2). The system we choose to implement based on this trade study is the fully-autonomous robotic system.

Table 3: System-Level Trade Study Numbers

Selection Criteria	**Weight Factor**	**a**	**b**	**c**
Capture rhythm of moving phantom liver	20	3	3	4
Track locations of tumors within the phantom liver	20	2	3	4
Estimate shape and size of embedded tumor	15	3	3	5
Provide visual feed of stiffness map to surgeon	15	2	4	4
Reduce cognitive overload of surgeon	10	2	3	4
Create point cloud of phantom liver	5	1	4	5
Determine where to palpate next	5	5	3	5
Duration of surgery	5	4	4	5
Hospitalization and recovery time	3	2	4	4
Cost incurred by the patient	2	3	5	5
Score (Max 5)		2.57	3.37	**4.42**

6.2 Sensing Subsystem Trade Study

We performed a subsystem-level trade study in order to determine the best sensors for our system. We considered having a: a stereo camera, b: a laser scanner (Blaser), or a&b: both. Table 4 shows the results of the trade study. The most important criteria on which we based our trade study include how autonomous the resulting system would be, the accuracy of the 3D model (M.P.5), and the simplicity of integration. These criteria were derived from our performance and non-functional requirements. Based on this trade study, we chose to use both the stereo camera and the Blaser.

Table 4: Sensor Subsystem Trade Study Numbers

Selection Criteria	Weight Factor	a	b	a&b
Autonomous (Active / Passive)	30	5	5	5
Accuracy of Point Cloud	30	3	3	5
Simplicity of Integration	25	5	5	3
Affordability	15	1	1	1
Score (Max 5)		3.8	3.8	**3.9**

6.3 Point Cloud Segmentation Trade Study

We performed a trade study on the algorithm to segment the organ of interest, i.e., the liver from the point cloud of all visible organs. The main criterion for evaluating this algorithm was the accuracy of segmentation in both well-lit and dimly-lit surgical settings (M.P.4). The 5 segmentation algorithms that we considered were a: Edge-based, b: Region-growing, c: Model-fitting, d: Unsupervised clustering, and e: Color-based, as shown in Table 5. Although we had initially thought that deep learning-based approaches would give us the best results based on prior knowledge, we first implemented a simpler HSV color-based segmentation based on this study and this happened to work well for our use case.

Table 5: Point Cloud Segmentation Algorithm Trade Study Numbers

Selection Criteria	Weight Factor	a	b	c	d	e
Accuracy	30	3	3	4	4	5
Need for apriori reference model	20	5	3	2	5	5
Computational efficiency	20	3	4	4	4	4
Robustness	20	3	3	4	4	5
Memory required for processing	5	4	3	3	4	3
Ease of use on a different organ	5	5	5	5	3	4
Score (Max 5)		3.55	3.30	3.60	4.15	**4.65**

6.4 Movement Simulation Platform Trade Study

This trade study was performed to determine what movement simulation platform (MSP) we would use to move our organs. There were 5 main options, 4 commercial(b: Mikrolar P1000, c: Symetrie Notus, d: Cemec Rotapod, e: 6 Axis Hexapod) and 1 self-built(a: Prof. Ken Goldberg self-built platform) which are evaluated in Table 6. The main criteria for this trade study were derived from how realistic the breathing motion would be and how feasible it would be to purchase (M.N.6), which are reflections of our requirements. Based on this trade study, we concluded that the best option was to build our own platform based on Prof. Ken Goldberg's design[5] since we could procure all necessary components with a fraction of the MRSD Capstone budget, and the resultant MSP would perform just as well as an off-the-shelf alternative.

Table 6: MSP Subsystem Trade Study Numbers

Selection Criteria	Weight Factor	a	b	c	d	e
Cost	20	5	1	1	1	1
Time	15	1	5	5	5	5
Capability	15	5	2	2	3	2
Difficulties	15	3	1	1	1	1
Accuracy	15	2	4	5	4	5
Linear Travel Range	5	2	4	5	3	3
Angular Travel Range	5	3	5	5	5	5
Size	5	5	1	3	4	2
Payload Capacity	5	2	4	5	5	2
Score (Max 5)		**3.25**	2.7	3.05	3	2.75

6.5 Motion Compensation Trade Study

We performed a trade study on the algorithm to compensate for liver motion. We found 2 main options, a: master-slave motion controller and b: resolved-rate motion controller, which are the first and second options in Table 7 respectively. We based the criteria for evaluating this algorithm on the accuracy of the motion-compensated tumor localization (M.P.9, M.P.10) and its compatibility with our simulation environment. The master-slave controller would require the use of a high-speed camera to track target points on the liver's surface while the resolved-rate controller would allow the robot to compensate for the liver's motion by using the predicted frequency alone. Since we did not create a stereo camera in simulation, we went ahead with the resolved-rate motion controller and were able to successfully compensate for the liver's motion.

Table 7: Motion Compensation Algorithm Trade Study Numbers

Selection Criteria	Weight Factor	a	b
Compatibility with simulation environment	30	1	5
Ability to process continuous image sequence	10	5	1
Prediction accuracy	30	1	3
Robustness to drastic organ movement	15	2	4
Robustness when organ image is not complete	15	3	1
Score (Max 5)		1.85	**3.25**

7 Cyberphysical Architecture

7.1 Cyberphysical Architecture Explained

The cyberphysical architecture in Figure 3 further demonstrates the functionality of our robotic surgical system, including methodology and the flow of data in more detail. It is divided into three main subsystems: the sensing subsystem, the data processing & organ modeling subsystem, and the searching & palpating subsystem. Again, lines in purple represent simulated inputs that we do not expect to have in the real system but that were required to run the simulation.

The sensing subsystem is responsible for taking visual and force data of the organ scene moving on the movement simulation platform (MSP). This signal information

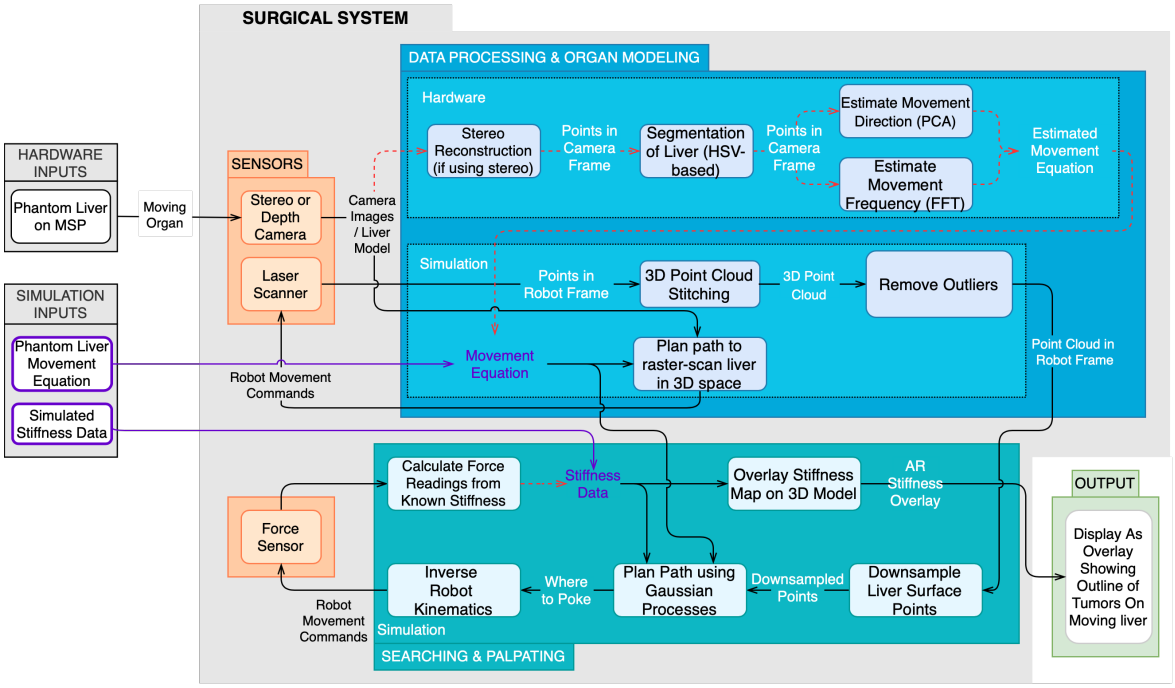


Figure 3: Chopsticks Surgical System Cyberphysical Architecture

is passed into the data processing & organ modeling subsystem and the searching & palpating subsystem.

The data processing & organ modeling subsystem will take in raw sensor data and transform it all into the camera-frame. The camera liver data will be segmented from all the visual data and passed on to calculate the movement equation of the organ. The movement equation and the visual data are used to help plan a path to scan the moving liver. The scanned liver points form a model of the organ. This model is passed to the searching & palpating subsystem.

The searching & palpating subsystem will perform a search on this model to find regions of high stiffness which correspond to embedded tumors. The search algorithm will calculate a location to poke and direct the force sensor to palpate there. After converting these force readings into stiffness data, the search algorithm will make an informed decision about where to poke next to gain the most information. Once the poking procedure is complete, the system overlays the stiffness information onto the 3D model and renders it onto the surgeon's camera feed.

7.2 Software Flowchart

Figure 4 describes the flow of software control through the system in greater detail. This flowchart was made keeping in mind the original vision for the project. We have had to make a few changes and assumptions to this flowchart since switching up to simulation.

The RealSense camera outputs a continuous stream of RGB-D data of the robot's workspace that is an input to the Segmentation module. An HSV mask that corre-

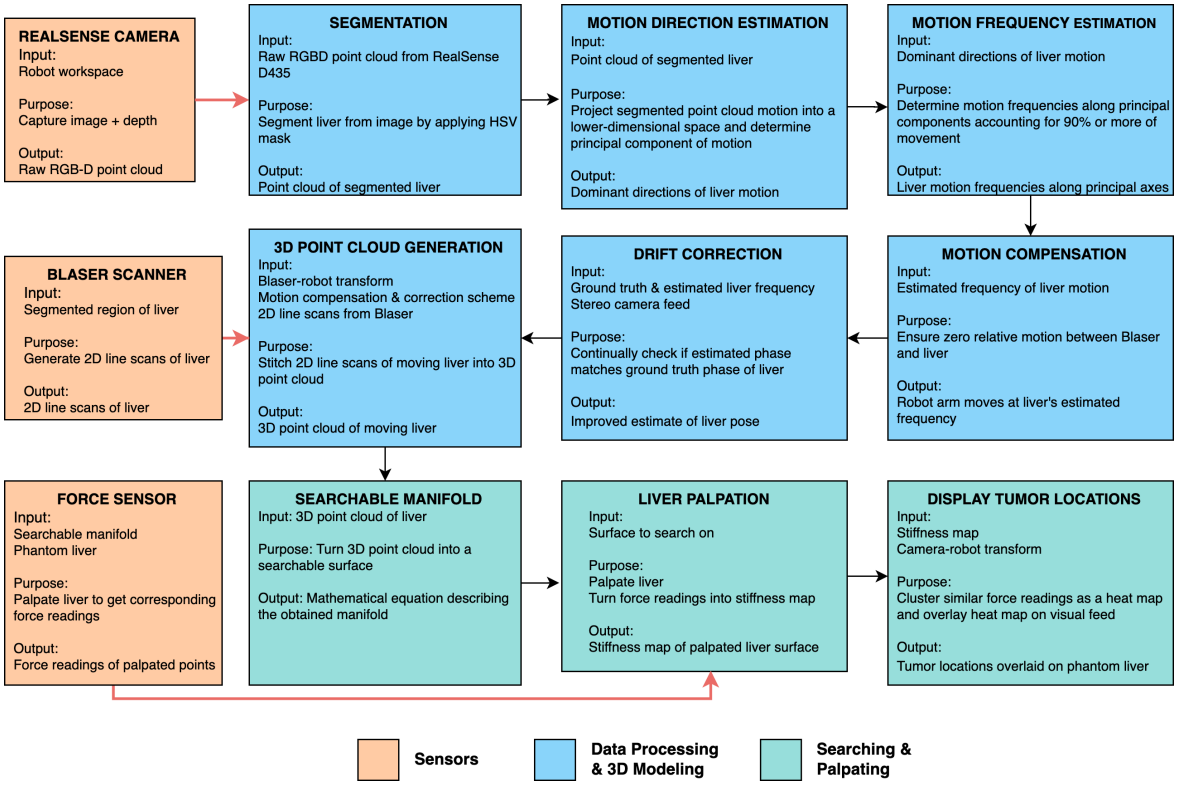


Figure 4: Software Components Flowchart

sponds to the liver's oxblood hue is applied to this data to segment out the liver points. The Motion Estimation module tracks the point cloud of the segmented liver over time to determine the liver's dominant direction of motion and the corresponding frequency. The estimate feeds into the Motion Compensation module, which determines the parameters of a closed-loop strategy to enforce zero relative motion between the liver and the robot arm. To offset an error in phase and to prevent the accumulation of drift in liver pose over time, a state estimation algorithm such as the Extended Kalman Filter would have to be deployed. We assumed that our system had perfect knowledge of the ground truth frequency and hence did not implement a drift correction algorithm in simulation. The motion compensation scheme is incorporated into the dVRK control code for the Blaser to scan the liver and generate a point cloud.

The Searchable Manifold module turns this point cloud into a searchable surface so that PSM2 of the dVRK can palpate the liver with respect to this model. The manifold's surface description is an input into the Liver Palpation module, which plans a motion-compensated trajectory for the Patient Side Manipulator 2 (PSM2) to search for embedded tumors using the force sensor mounted on its end-effector. We did not simulate a force sensor in simulation; we faked tumors on the liver CAD file and followed a sampling-based approach to palpate the liver's surface. The palpated locations and force readings are paired together in the robot-frame and fed into the Display module, which uses the camera-robot transform to render the stiffness information as a heat-map on the surgeon's visual feed.

8 System and Subsystem Descriptions

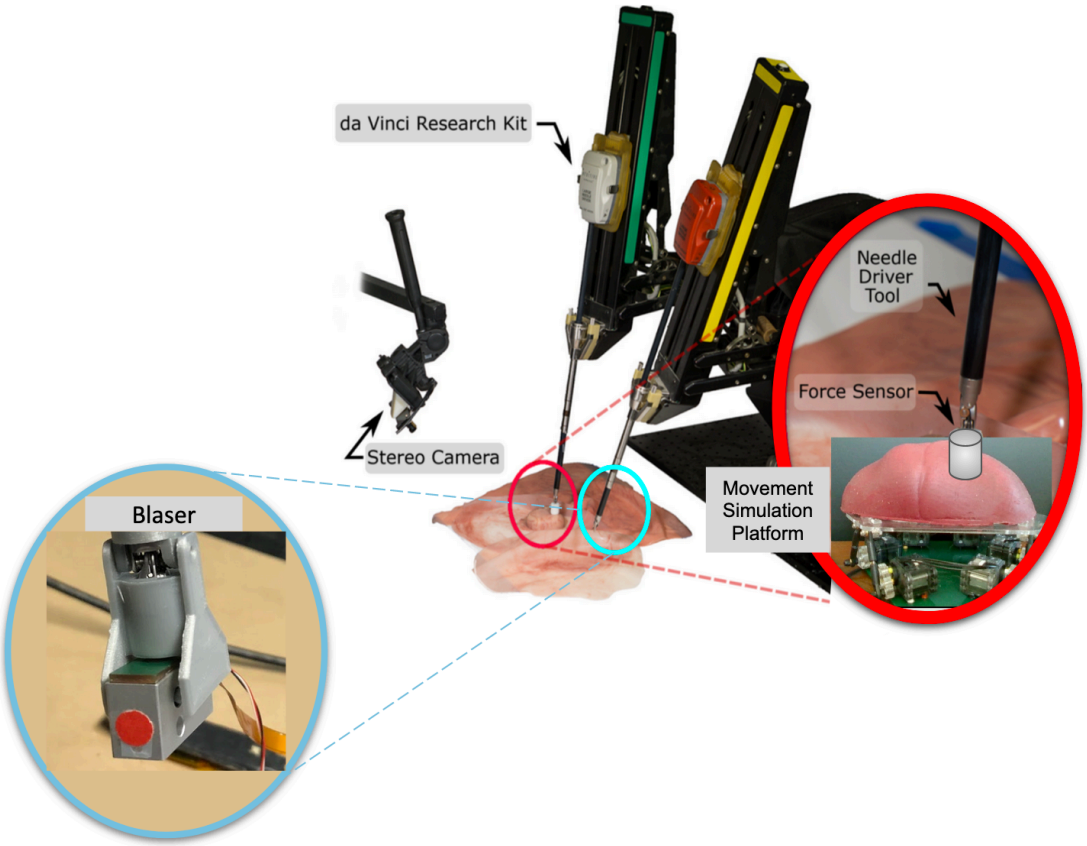


Figure 5: Chopsticks Surgical System Depiction

8.1 Requirements in Relation to Subsystem

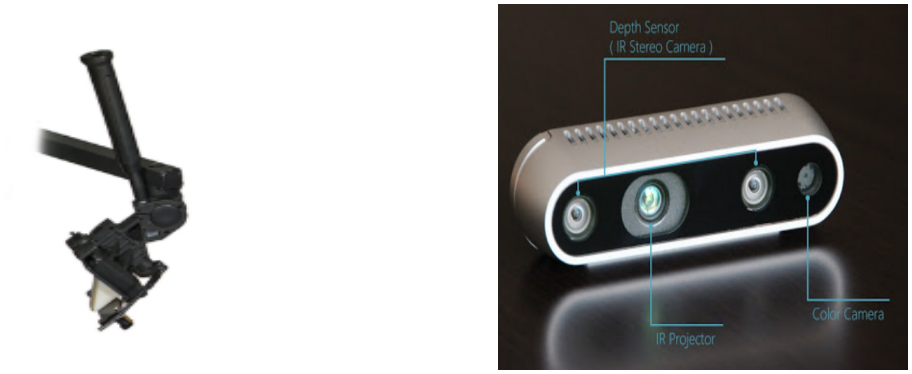
As shown in Figures 2 and 5, our system is composed of four main subsystems: sensors, movement simulation platform, data processing & organ modeling, and searching & palpating.

The sensors subsystem includes the stereo camera, the Blaser sensor, and the force sensor (M.N.3). The MSP subsystem includes the MSP and the phantom organs. The data processing & organ modeling subsystem includes the algorithms related to frame registration (M.P.1 and M.P.2), point cloud construction (M.P.3 and M.P.5), image segmentation (M.P.4), and motion estimation (M.P.6). The searching & palpating subsystem includes algorithms related to searchable surface transformation, searching trajectory planning (M.P.7), force reading, stiffness data generation (M.P.9 and M.P.10), inverse robot kinematics, and motion compensation. M.P.8 refers to the entire system time performance.

8.2 Sensors Subsystem

The sensors subsystem is composed of a stereo camera, a Blaser, and a force sensor. The goal of this subsystem is to capture sufficient data for the data processing & organ modeling subsystem to estimate organ motion.

8.2.1 Camera



(a) **ELP Stereo Camera** (b) **Intel RealSense Depth Camera D435i**

Figure 6: Two Different Cameras used So Far

The camera will image the phantom liver. The location of corresponding 3D points is obtained by stereo reconstruction of the images or directly from the depth sensor and color camera. The ELP stereo camera (model 1MP2CAM001), shown in Figure 6a, was registered to the dVRK. However, in light of the COVID-19 pandemic, all project work in the second half of the Spring semester had to be performed remotely. Since we could not move the ELP stereo camera from the Biorobotics Lab, an Intel RealSense Depth Camera D435i, shown in Figure 6b, was borrowed from inventory. The RealSense D435i was used to perform the tasks of segmentation, tracking, and frequency estimation of the phantom liver.

8.2.2 Blaser Sensor: Hardware

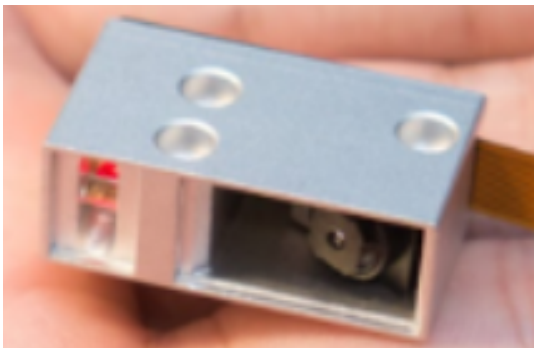
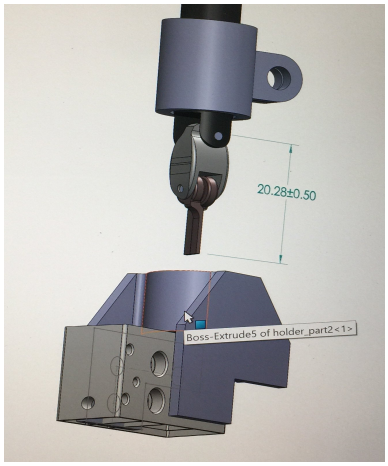
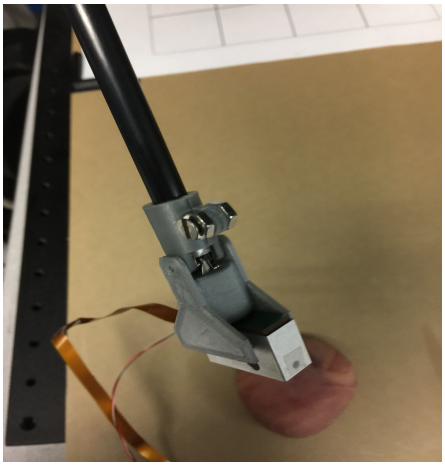


Figure 7: Blaser Closeup

The Blaser sensor, shown in Figure 7, was developed by the Boeing-CMU team for short-range high-precision perception applications. It is a small and light-weight 2D laser scanner that is particularly well-suited for confined space applications since it has a camera situated in proximity to its diode. The Blaser is mounted on Patient Side Manipulator 1 (PSM1) of the dVRK through a custom-designed holder, shown in Figure 8b, which guarantees that the Blaser can move seamlessly with the wrist of the arm. The holder was created in SolidWorks, and its CAD model is shown in Figure 8a. The Blaser-frame and the robot-frame are registered, allowing for the 2D line scans generated by the Blaser to be stitched into point clouds.



(a) **Blaser Holder CAD Model**



(b) **Blaser Mounted to Wrist through Holder**

Figure 8: Custom-designed Blaser Holder

8.2.3 Blaser Sensor: Simulation

Due to the COVID-19 pandemic, we moved our system entirely to simulation. We coded the Blaser simulator using the Python package '`vtk`' (https://pypi.org/project/vtk/). The Blaser sensor is modeled as a point on the robot arm's end-effector that shoots out laser beams. The angles at which these beams bounce off of an obstacle are used to determine how far that obstacle is from the simulated Blaser. We started by converting the liver CAD file into an Oriented Bounding Box (OBB) Tree. The Blaser shoots a fixed number of rays (vtk vectors) from the robot's end-effector on which it is to be mounted. We used '`IntersectWithLine`' function to retrieve the first collision along each line. Figure 9 demonstrates the principle of the Blaser simulator. By calculating the distance between the starting point (red dot in Figure 9) and the collisions (green dots in Figure 9) for each beam, we can estimate the location of each collision in the world coordinate frame. We tuned the spanning angle, the number of total beam arrays, and the noise range to optimize the results and to be able to run on our computers in the simulation without too much computational effort.

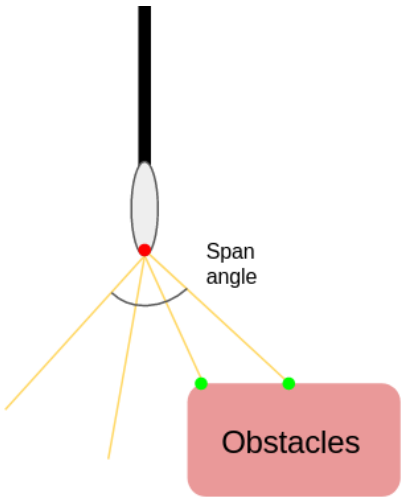


Figure 9: Blaser Sensor Simulation. Red dot: starting point; Green dots: collision points.

8.2.4 Force Sensor

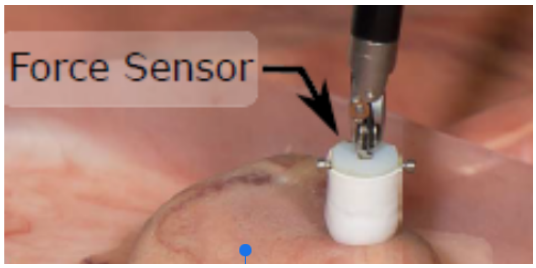


Figure 10: Force Sensor[4]

After several team meetings, the team decided not to simulate a force sensor. In this section, we describe the force sensor that we had planned to use on hardware in Spring 2020 followed by an explanation of the simple method we used in Fall 2020 to replace the force sensor.

The original plan was to use a miniature force sensor developed by the Biorobotics lab at CMU [4]. It is a force-sensitive resistor-based (FSR) force-to-voltage transducer operating in thru-mode electrodes configuration. It was to be connected to the Patient Side Manipulator 2 (PSM2) of the dVRK via the needle driver tool and sense contact forces. Coordinates are inputted to the PSM2 to guide it to where the force sensor should palpate. The sensor outputs force readings that are processed to obtain a stiffness distribution across the organ. An image of the sensor is shown in Figure 10.

In Fall 2020, we decided not to simulate a force sensor. We simulate tumors on the liver's surface by picking arbitrary points to act as tumor centers and assigning high stiffness values to the tissue surrounding those points. When the robot visits a point on the liver's surface during the palpation procedure, the system returns the stiffness value at that point. If we did not previously assign the palpated location a stiffness value in the ground truth stiffness map, the sampling returns the stiffness value of the closest point in the ground truth. Noise sampled from a uniform distribution is added to the returned values.

8.3 Movement Simulation Platform Subsystem

The purpose of the MSP is to simulate the motion pattern of the liver due to a patient's respiration and heartbeat rhythm during surgery. The MSP was constructed by emulating the work of Vatsal Patel et al. as described in [5]. The MSP has 6 degrees of freedom (DOF) to move and rotate in 3D space. The MSP comprises two main subsystems, hardware and software subsystems. The components are listed in Table 8, and the building process will be discussed in the following section.

The main structure of the MSP is made of acrylic. We used a laser cutter to manufacture desired pieces according to the work of Vatsal Patel et al. [5] with some modifications to fit our project. In order to provide more stability during operation, additional adhesive methods were added, such as superglue. Some more modifications

Table 8: Parts for the MSP

Component	Quantity	Purpose
XL320 Servo Motor	6	Actuate the platform
OpenCM 9.04-C Board	1	On board microcontroller
Acrylic Parts	2	The main structure of the platform
Screws and Nuts	25	Connect components together

include reducing the tolerance between assembled parts and additional structural support. The final MSP product is shown in Figure 11.

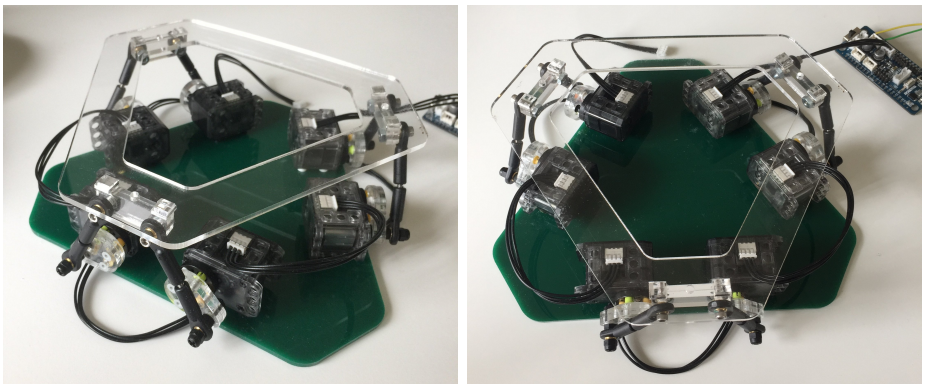


Figure 11: Movement Simulation Platform

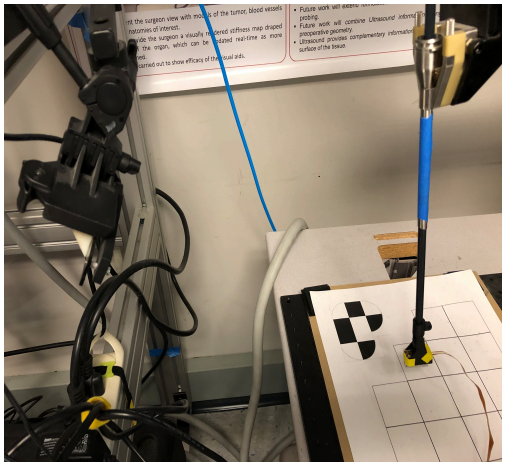
The software we used to program the MSP is ROBOTIS OpenCM API, and C language. We control these XL320 servos differently from general servo motors. Firstly, all servos were connected in series to the microcontroller. Therefore, in order to control each servo, the commands are given in the form of [ServoID, Command Type, Command Goal].

8.4 Data Processing & Organ Modeling Subsystem

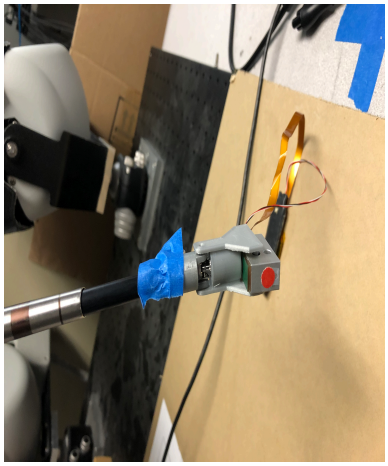
The data processing & organ modeling subsystem includes all algorithms of preprocessing before searching & palpating the organ, including those for frame registration, point cloud reconstruction, and organ motion estimation. The subsystem will take points from the Blaser scanner, images from the stereo camera and points from robot markers as input, and output a moving 3D model of the organ in the camera reference frame for further processing by the searching & palpating system. The algorithms within this subsystem will be explained in greater detail in the subsections below.

8.4.1 Frame Registration Algorithms

We use registration algorithms to get full knowledge of transformations between the camera and robot, and the Blaser and robot. The transformations allow us to fuse inputs from different sensors from different frames. The final outputs can also then be visualized in the camera-frame and displayed to the surgeon on the visual feed [1].



(a) **Stereo Camera and Robot Setup**

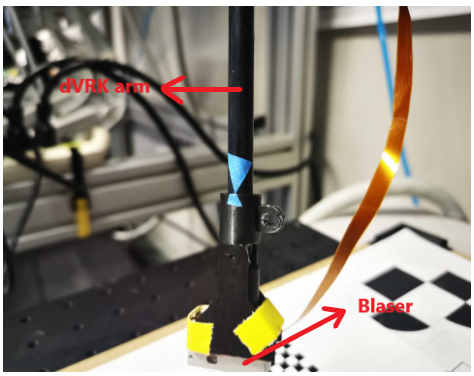


(b) **Orange Marker on Blaser**

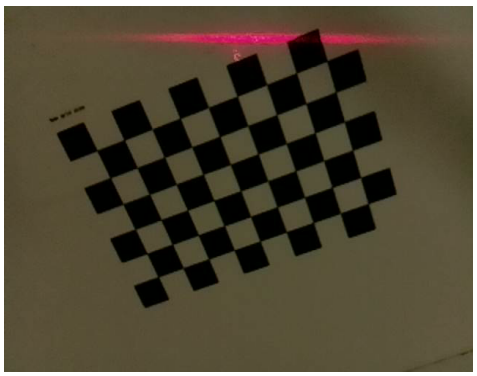
Figure 12: Camera-Robot Registration Setup

An ELP stereo camera overlooks the workspace of the robot, as shown in Figure 12a. The rigid body transformation between the camera-frame and the robot-frame was obtained using Horn's method. As shown in Figure 12b, we stuck an orange marker on the Blaser with a known offset with respect to the robot's wrist. The left and right images of the stereo camera were processed to find the centroid of the orange marker, which gave its coordinates in the camera's frame. We moved the robot to a fixed set of six points that were chosen to cover a substantial amount of the robot's workspace. Horn's method was then used to optimize for the best homogeneous transformation matrix that relates the robot-frame and the camera-frame.

We used a checkerboard to register the Blaser-frame and the robot-frame. Sixty images of the checkerboard were obtained by manually moving the robot arm to different positions in the workspace and orienting the wrist in different directions to capture as much movement of the robot's wrist as possible. Figures 13a and 13b show the Blaser mounted on the robot arm and the checkerboard pattern respectively. The 6DOF pose of the Blaser that resulted in each image was estimated using the perspective-n-point (PnP) algorithm. Once the pose of the Blaser was estimated for each image, we performed bundle adjustment to obtain the transformation matrix between the robot's frame and the Blaser's frame.



(a) **Blaser Mounted on Wrist**



(b) **Checkerboard used for Registration**

Figure 13: Blaser-Robot Registration Setup

8.4.2 Point Cloud Reconstruction Algorithm

We mounted the Blaser on the Patient Side Manipulator (PSM1) of the dVRK and programmed it to move across the phantom liver in a raster scan-like fashion. The Blaser sensor captures 2D line scans of the liver, which the stitching code uses to generate a point cloud. The stitching code converts 2D line scans obtained from the Blaser into a point cloud using the transformation matrix that relates the Blaser-frame and robot-frame.

The 4x4 transformation matrix of the Blaser-robot registration is loaded from a YAML file. A ROS tf broadcaster adds the Blaser frame to the robot arm tf tree. There is one subscriber to read the current Blaser-to-World transformation, and another to read the real-time points published by the Blaser. The dot product of these two subscribers' input is stored in a list at each timestep. Rviz provides real-time visualization of the 2D line scan obtained at each timestep being "stitched" into the growing point cloud. This process works similarly in the simulation. An example of this is shown in Figure 24 later in the paper for different liver shapes.

8.4.3 Image Segmentation Algorithms

The segmentation algorithm displays only phantom liver pixels from the pixels of all visible abdominal organs such as the liver, kidneys, stomach, etc. This algorithm takes in the camera's images and output only points of the liver, in preparation for the searching & palpating subsystem. The method we implemented was an HSV-based segmentation. This method looks for pixels within the specified hue, saturation, and value ranges and classifies those as part of the liver. Though this method is sensitive to other similarly colored objects in the background, the environment that we test in and the body in general are pretty controlled environments that would not have similarly colored objects. Figure 14 shows the segmentation algorithm working.

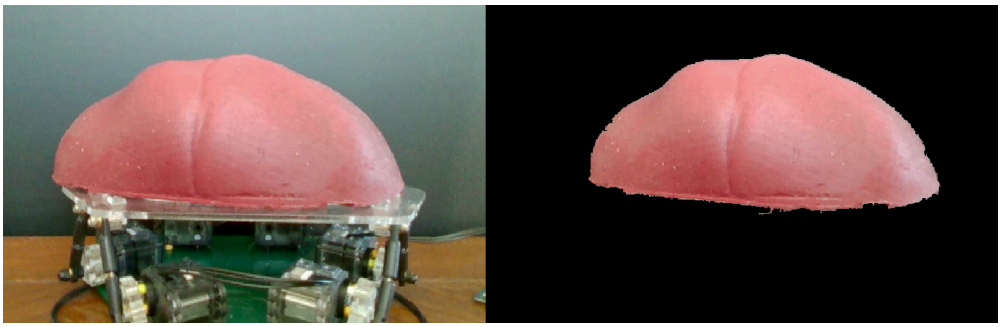


Figure 14: Results of Image Segmentation Algorithm in Brighter Light

8.4.4 Motion Estimation Algorithms

The motion estimation algorithms in [6], [7] estimate the direction along which the organ exhibits the most movement, as well as the frequency of this movement using the visual data from the camera over time. The former will be estimated using Principal Component Analysis (PCA) and the latter using Fast Fourier Transform (FFT).

In the robot's workspace, the camera focuses on the moving liver to capture its movements across time. A motion compensation scheme can only be devised if the main direction and dominant frequency of motion of the liver are known. The camera observes the motion of the liver in its coordinate frame, but the motion is often not aligned with the axes of the camera. In Figure 15, the MSP is translating along the world-frame's z-axis, but the camera overlooks its movement from approximately a 45° perspective; therefore, the translatory motion of the liver will be resolved along the two axes of the camera and described in the camera's frame. This is followed by applying Principal Component Analysis (PCA) to project the motion onto a low-dimensional dominant-direction space to yield a projection matrix. Fast Fourier Transform (FFT) is applied to the resultant principal components of motion to analyze their frequency spectrum and obtain the values corresponding to the principal components.

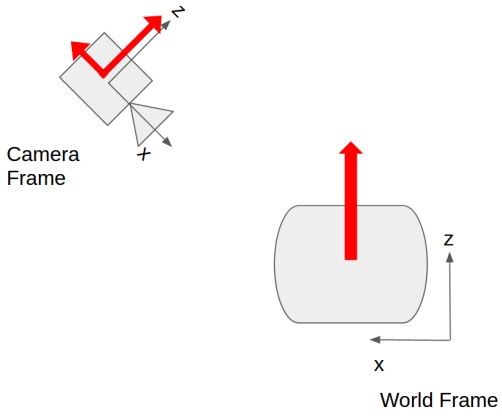


Figure 15: Motion Decomposition in Two Frames

8.4.5 Motion Compensation algorithm

In real surgery, due to the patient's breathing pattern, the abdominal organs would move periodically. Once the frequency of this motion is estimated, it has to be compensated for by our system. Unless the estimated frequency and the compensation frequency are identical, drift in the phase of the organ will accumulate over time and require correction. However, in our case, we assume that our system has exact knowledge of the ground truth frequency and compensate for it without the need for a state estimation algorithm. Our motion compensation algorithm, called the resolved-rate motion control [9], interleaves prediction and pursuit to ensure no relative motion exists between the robot and the liver. To visit a point P on the liver's surface (shown in red in Figure 16), the controller first predicts where point P is in the current time step using the ground truth frequency and amplitude of the liver's motion. If point P is farther away from the robot's end-effector than the permitted step size, the robot is moved to a waypoint (shown in green in Figure 16) along the vector joining the robot end-effector and P. This process repeats over a few time steps as the end-effector closes in on P and eventually reaches it. Using the resolved-rate motion controller, the robot moves smoothly between waypoints, and the vertical offset added before each transition ensures that the robot arm seldom collides with the liver.

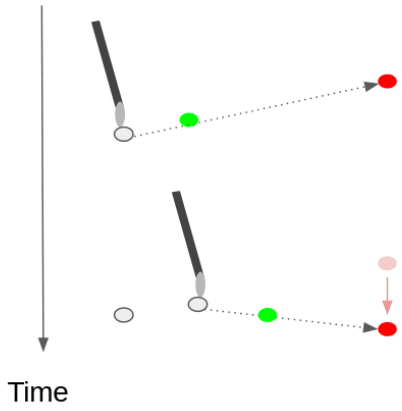


Figure 16: Motion Compensation

8.4.6 Blaser Scanning

To obtain a 3D point cloud of the liver, we used the Blaser to scan the entire liver in 2D and combined these scans. We imported the dVRK robot model from John Hopkins University's open-source GitHub repository to our Rviz environment and attached our Blaser simulation unit to the robot end-effector. We sampled a user-defined number of points on the surface of the liver (detailed in the Data Processing section) and defined an offset D, which is the distance between the robot end-effector and the liver surface when the robot visits each sampled point along the corresponding surface normal vector. The motion compensation algorithm ensures that the distance between the robot end-effector and the liver surface is D through the scanning procedure. We noticed that the Blaser simulator was eating into our processing power if it stayed switched on the entire time that the motion compensation algorithm tried to close in on the destination point. Therefore, we modified the Blaser simulation to switch on and shoot out laser beams only when the robot end-effector reached the destination point. The Blaser scanning the liver in our simulation environment is shown in Figure 17. The red object in Figure 9 is the OBB tree representation of the liver, and the green line on the liver's surface are points where the laser beams collide with it.

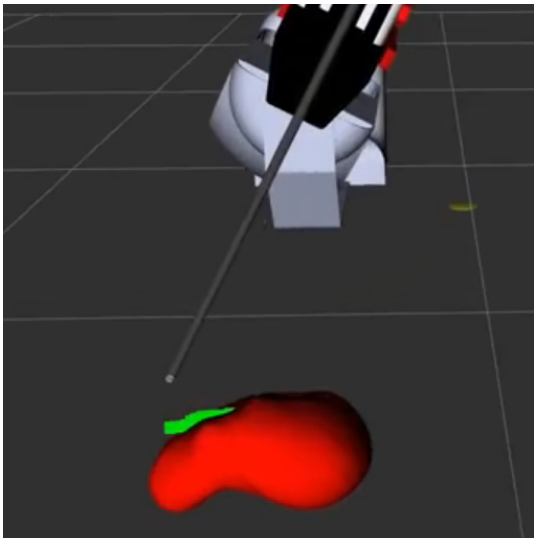


Figure 17: Blaser scanning in simulation

8.5 Searching & Palpating Subsystem

This subsystem includes all the algorithms necessary to take a 3D model of an organ and output a stiffness map. It includes algorithms that transform the data to the correct form, search over the organ to detect tumors, control robot kinematics, and render the stiffness map. Figure 5 shows the dVRK robot that the Chopsticks Surgical System will use palpating the silicone model of an organ.

8.5.1 Searchable Surface Transformation

This subsystem turns blaser scanning result to an executable path for robot to palpate the liver. Figure shown below is the intuitive explanation of it.

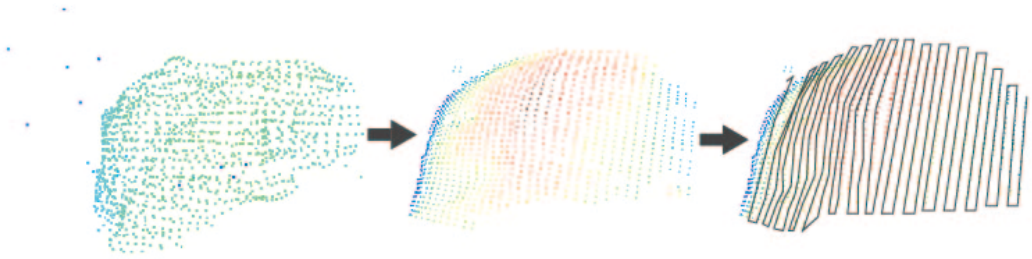


Figure 18: Example of Point Cloud transformed into Searchable Surface

The point cloud obtained from the Blaser is transformed into a searchable surface through a sequence of steps, shown in Figure 18. The image on the left shows the raw Blaser output containing outliers and noise. The image in the middle is the filtered point cloud obtained by performing two processes: outlier removal and voxel grid filtering. In outlier removal, points that have fewer than five neighbors within a radius of 0.01 meters are removed from the data. Voxel grid filtering attempts to smoothen the resultant point cloud. A new point whose normal and position is the average of its neighboring points is added to densify the point cloud, where necessary. The next step is transforming the augmented point cloud to an executable path for the robot, as shown in the rightmost image. The points in the cloud are then reordered to generate a snake-shaped path for the robot to use during the palpation procedure.

8.5.2 Searching Trajectory Planning for Palpation

We introduced two methods to palpate our liver in simulation. The first is what we called the "naive" method and the second is the Gaussian Processes-based palpation. The naive method creates palpation trajectories in a way similar to the scanning process. The Gaussian Process method uses a Gaussian distribution to intelligently sample points and palpates the liver. A description of the two methods follows.

Naive Palpation: After the scanning procedure, the obtained point cloud of the liver is post-processed to generate an array of points on the liver's surface that acts as a raster scan-like trajectory for the robot arm to follow during the subsequent palpation procedure. The naive palpation controller commands the robot to visit all of these locations. However, it would take the robot over 20 minutes to visit all of these points. Therefore, we optimize the palpation by skipping points in the "soft" regions.

We defined a stiffness threshold of K and set a local variable N that kept track of the number of consecutive points that have stiffness values under the threshold K. A soft region is one where N is less than a predefined threshold of C, so the robot can skip ahead in its palpation and save some time. On the other hand, if the robot is in a stiff region, it visits every point in the trajectory to maximize its chances of accurately localizing the tumor. The values K, N, C were set by trial and error. Different values were tested to strike a balance between accuracy and time.

Gaussian Process Palpation: After getting the point cloud of the liver through the scanning procedure, we project the 3D points into a 2D plane and overlay a grid over it. The density of the grid is adjustable according to the point cloud density. We then apply Gaussian processes for palpation and stiffness retrieval over the scale of the grid. The whole process consists of two stages.

In the first stage, the controller randomly chooses N points to poke through stratified sampling. This stage balances exploration and exploitation. When there are multiple tumors in the organ, the arm may poke a high stiffness area of one tumor at the beginning of the palpation process. Then the Gaussian Process would get stuck in a local optimum and continue to poke only that tumor, neglecting other tumors. The prior knowledge from stratified sampling solves the problem.

In the second stage, we utilize Gaussian Processes (GP) to model the distribution of stiffness on the organ. By using GP, we assume a smooth change in the stiffness distribution across the organ. Since every point on the organ's surface can be mapped in a 2D grid, the domain of search used is $X \subset \mathbb{R}^2$. The measured force and position after probing the organ by the robot at x provide the stiffness estimation represented by y.

A GP is defined by its mean and covariance functions f_{GP} and k respectively. Given a d -dimensional search domain $X \subset \mathbb{R}^{\mathrm{d}}$, the distribution of stiffness values at a point $\boldsymbol{x} \in X$ is represented by a random variable, y, and has a Gaussian distribution, $N\left(f_{GP}(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\right)$ where we abbreviate $\sigma^2\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x})$. Given a set of n stiffness observations $\bar{y} = [y_1, y_2, \ldots, y_n]^T$ at $\bar{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T$, GP regression is used to make predictions on the distribution of stiffness values at a new point $\boldsymbol{x}_* \in X$ on the liver surface using:

$$p\left(y_* \mid \overline{\mathbf{y}}\right) \sim N\left(\boldsymbol{K}_* \boldsymbol{K}^{-1} \overline{\boldsymbol{y}}, k_{**} - \boldsymbol{K}_* \boldsymbol{K}^{-1} \boldsymbol{K}_*^{\boldsymbol{T}}\right)$$

where $\boldsymbol{K}$ is the $n \times n$ covariance matrix whose elements $\boldsymbol{K}_{ij}(i, j \in [1, \ldots, n])$ are calculated using any positive definite covariance function $k\left(x_i, x_j\right)$ (we use the squared exponential covariance function). Similarly, $\boldsymbol{K}_*$ is a $1 \times n$ vector defined as $\boldsymbol{K}_* = \left[k\left(\boldsymbol{x}_*, \boldsymbol{x}_1\right), \ldots, k\left(\boldsymbol{x}_*, \boldsymbol{x}_n\right)\right]$, and finally $k_{**} = k\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right)$.

Then based on the stiffness estimation predicted by the set of Gaussian Processes, we rank all the points and decide to poke the location with the highest estimated stiffness, since high stiffness usually corresponds to tumor areas. After each poke, the Gaussian Process will update its estimate, leading to an automatic iterative process. This intelligent searching method will help reduce palpation times on softer areas and

speed up the tumor localization process.

Coming to the acquisition function used for ranking and searching, we tried different ones, including Active Learning and Bayesian optimization (BO).

1) Active Level Set Estimation: This algorithm determines the set of points, for which an unknown function (stiffness map in our case) takes value above or below some given threshold level h. LSE guides both sampling and classification based on GP-derived confidence bounds. The mean and covariance of the GP can be used to define a confidence interval,

$$Q_t(\boldsymbol{x}) = \left[f_{GP_t}(\boldsymbol{x}) \pm \beta^{1/2}\sigma_t(\boldsymbol{x})\right]$$

for each point $x \in \bar{X}$, where the subscript t refers to time. Furthermore, a confidence region C_t which results from intersecting successive confidence intervals can be defined as,

$$C_t(\boldsymbol{x}) = \bigcap_{i=1}^{t} Q_i(\boldsymbol{x})$$

LSE then defines a measure of classification ambiguity $a_t(\boldsymbol{x})$ defined as,

$$a_t(\boldsymbol{x}) = \min\left\{\max\left(C_t(\boldsymbol{x})\right) - h, h - \min\left(C_t(\boldsymbol{x})\right)\right\}$$

LSE chooses sequentially queries (probes) at $\boldsymbol{x}_*$ such that,

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}\in X} a_t(\boldsymbol{x})$$

2) Bayesian Optimization: In addition to the above active learning algorithms, we also tried Bayesian optimization algorithm (BO). BO is a sequential sampling strategy for finding the global maxima of black-box functions. A GP is used as a surrogate for the function to be optimized. BOA uses the posterior mean, $\mu(\boldsymbol{x})$, and variance, $\sigma^2(\boldsymbol{x})$, of the GP for all $\boldsymbol{x} \in X$, to sequentially select the next best sample as the point that maximizes an objective function such as the Expected Improvement (EI).

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}\in X} EI(\boldsymbol{x})$$

$$EI(\boldsymbol{x}) = \begin{cases} \left(f_{GP}(\boldsymbol{x}) - y^+\right)\Phi(z) + \sigma(\boldsymbol{x})\phi(z) & \text{if } \sigma(\boldsymbol{x}) > 0 \\ 0 & \text{if } \sigma(\boldsymbol{x}) = 0 \end{cases}$$

where $z = \left(\frac{f_{GP}(x) - y^+}{\sigma(x)}\right)$, y^+ is the current maximum. $\phi(\cdot)$ and $\Phi(\cdot)$ are the probability density function and cumulative distribution function of the standard normal distribution, respectively.

8.5.3 Stiffness Data Retrieval

Our original plan for the project on the hardware involved mounting a miniature force sensor on the end-effector of one arm of the robot. The system would receive high force readings at points on the liver's surface that contained embedded tumors while the soft, "healthy" portions of the phantom liver would return low force readings. The search algorithm would decide where to poke next based on the force readings. At

the end of the palpation procedure, using a multivariate Gaussian distribution [8], the stiffness data would decide which stiffness values exceeded a threshold that indicates the presence of a tumor. The shape and location of tumors determined by the algorithm in the organ would be overlaid on the visual feed for the surgeon to see, as depicted in Figure 19.

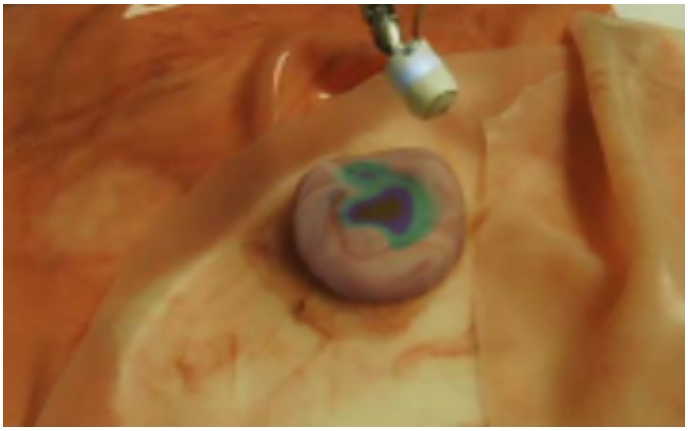


Figure 19: Overlaid Stiffness Data Provides Tumor Information[1]

When we switched our project to simulation, we modified the physics of the embedded tumors. For example, to "embed" one tumor in the liver, we arbitrarily picked a point on the liver's surface to act as the tumor center. The surrounding liver tissue within a certain Euclidean distance would make up the tumor centered at the chosen point, while regions farther away are considered to be healthy tissue. Thus, the ground truth "stiffness" at a point on the liver is inversely proportional to its distance from the tumor center.

Since we did not develop a force sensor in simulation, the robot cannot receive force readings by palpating the liver. Instead, when the robot palpates a point on the liver's surface, it is assigned the stiffness value of the point nearest to it in the ground truth tumor map. Although the tumor modeling was modified in the absence of a simulated force sensor, the principle of tumor localization through organ palpation remained unchanged.

8.5.4 Display

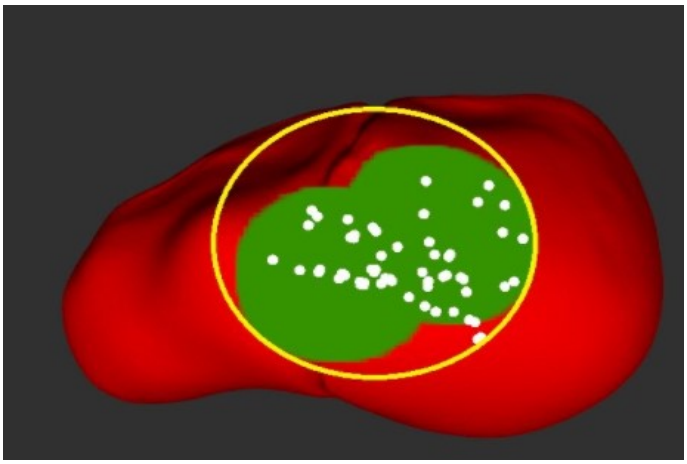


Figure 20: Palpated points(white) overlaid on ground truth tumor(green)

Once the robot completes the palpation procedure, a visualization of the results as shown in Figure 20 appears for the user/surgeon by switching on two ROS topics. The first topic publishes the shape and location of the ground truth tumor, shown in green, while the second topic publishes palpated points in white. During development, we overlay both visualizations on the liver to get an understanding of what percentage of the tumor(s) was classified correctly. The minimum bounding circle shown in yellow is sketched offline to show how we calculated our accuracy percentages.

8.6 Analysis and Testing

8.6.1 Sensors

On Blaser hardware, we manually tested the range of the Blaser sensor to be 5 - 7 cm by moving the Blaser towards and away from the surface of the liver and seeing when we still received reliable line scans. We also determined from this test that the Blaser had the best line scans when mostly aligned with the surface normal. These results have guided our design of the Blaser scanning algorithm.

In simulation, we tested the Blaser with different beam densities. In theory, a higher density is better for denser point cloud reconstruction. However, the density is also constrained by limited computational power. In FVD, we can tune the beam density from 5 beams per second to 100 beams per second. We also found that a density of 10 beams per second is the best for both efficiency and data quality.

For the simulated force sensor, we directed the robot to 100 assigned points and calculated the RMSE stiffness with respect to the ground truth stiffness. We got an RMSE = 0 with no noise added on. We also tested the sensor when adding uniformly distributed noise. By tuning the amplitude of the noise, the RMSE varied between 0 and 0.05.

8.6.2 Movement Simulation Platform

We tested each motor to ensure that they all received commands correctly. In the process of building the MSP, we accidentally broke four motors' servo IDs by assigning them an incorrect value and had to buy some new motors to replace them. By the SVD, we could command the MSP to exhibit the following motions: 1) 1D: up and down movement (Z-axis), 2) 2D: circular movement in XZ-plane 3) 3D: translational and rotational movements. We had discussions with a few doctors and determined that the liver movement in the human body is mostly one dimensional. Therefore, despite being able to program the MSP to move in 3D, the up-and-down motion will be used in the project to model liver motion.

8.6.3 Data Processing & Organ Modeling

We used a RealSense camera to read in the raw point cloud of the scene and segment out the liver based on its HSV threshold as shown in Figures 21 and 22. ICP is applied to get the rotation matrix R and translation vector $\mathbf{t}$ that relate the point cloud obtained at each timestep to the one at the previous timestep. The liver is tracked over time using the obtained 4x4 homogeneous transformation matrix. However, this method has some drawbacks: the displacement of the liver between two consecutive time steps is on the order of 0.1 mm, while the image noise is about ten times higher and tends to corrupt the tracking. A workaround is to simply center the data about its mean and track the centroid of the segmented liver. The results of this method are surprisingly robust and accurate, albeit rotational movements of the liver cannot be captured. This was not a problem since we considered only the 1D motion of the liver in our project. The Intersection over Union (IoU) that quantifies the segmentation accuracy was over 95% in all attempts.

Figure 21: Segmentation in brightly-lit conditions

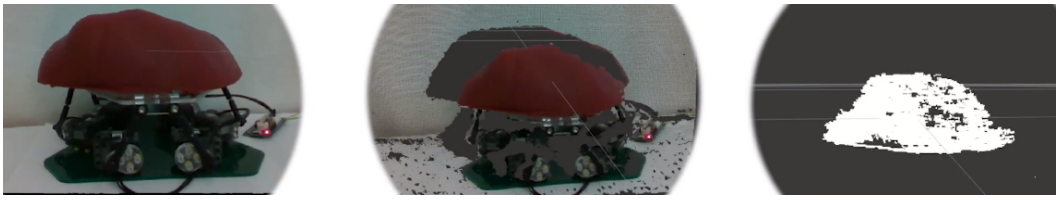


Figure 22: Segmentation in poorly-lit conditions

We also placed the stomach and the liver at different distances from the RealSense camera and measured the quality of segmentation. The highest accuracies are obtained

when the organs are placed 50 - 60 cm in front of the camera. Figure 23 is a view of the organs through the stereo camera at the Biorobotics Lab.

Figure 23: Modeling pipeline tested on hardware - red liver and yellow stomach on a plane

We tested our system in simulation on livers of different surface profiles. As seen in Figure 24, shapes 1 and 2 are rugged compared to the smooth profile of the liver we use. We ran the canning procedure on the rugged livers to show that the Blaser can capture nuanced surfaces as well. The result RMSE of the livers are more or less the same (6mm).

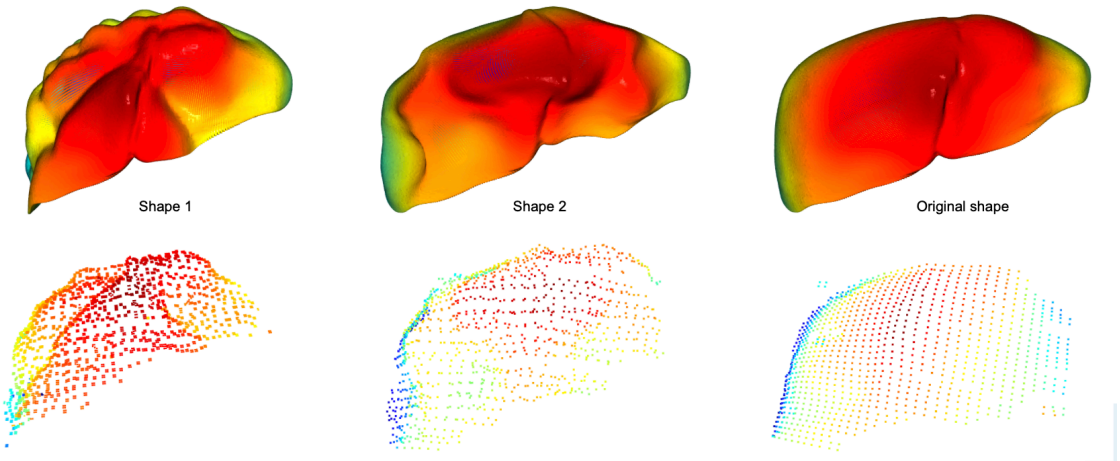


Figure 24: Modeling pipeline tested on three different shape of liver

8.6.4 Search and Palpation

We tested our search and palpation procedures in simulation on tumors of different numbers and shapes. We also tried different thresholds for the maximum number of times the robot is allowed to palpate the liver. As expected, the larger the number of palpated points allowed, the longer the robot takes to complete the procedure. The robot takes about 5 minutes and 8 seconds to palpate 50 points on the liver's surface, and this trend scales almost exactly linearly as the number of palpated points increases. The accuracy of tumor localization is 100% nearly all the time.

8.7 Performance at FVD

We conducted one main experiment for the Fall Validation Demonstration. This involved running our entire system altogether and measuring various outputs along the way. The first output we measured was in regards to the point cloud accuracy, M.P.6. We were able to generate a point cloud of the phantom liver with 6.34mm RMSE, which is within our requirements of 10mm. To calculate the RMSE, we fed in the ground truth point cloud of the liver that we had from the CAD model and the point cloud pcd file from the points that we scanned. We then manually align the two point clouds to give the algorithm a better starting point and it runs ICP to align them further. Figure 25 shows the output from the point cloud alignment software.

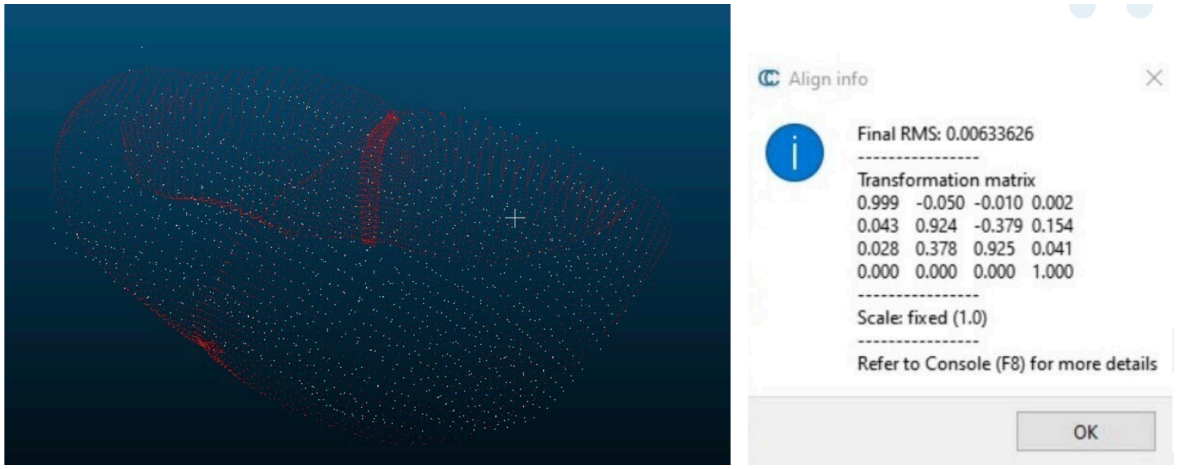


Figure 25: Point Cloud Comparison with Ground Truth Results

The second metric we measured was the time it took to complete each part of the procedure. This involved timing the scanning time, the palpation time, and the overall time taken, which encompasses software processing time as well. This section of metrics is related to M.P.5, M.P.7, and M.P.8. To measure these we checked the computer clock time before and after starting every procedure and subtracted these in the code to get a final time. The results of doing these timings in a situation with one tumor without any noise in the stiffness are shown in Figure 26. However, if we add more noise to the palpation procedure as we did in FVD encore, we got the results shown in Figure 27. Without the noise, we meet our desired performance requirements for total procedure time and are 27 seconds off for our palpation time. We meet our mandatory performance requirements for palpation and scanning time. If we do add more noise, we still meet all our mandatory performance requirements. This showed that our system was relatively robust to noise.

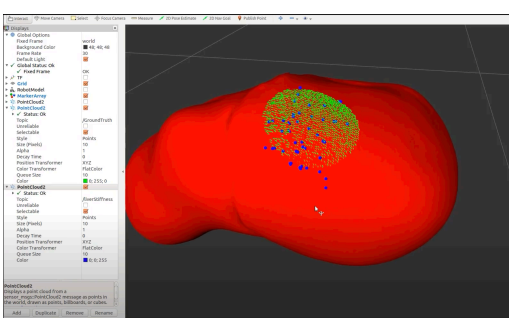
```
Time taken to 3D-scan the liver = 3 minutes, 26 seconds
Time taken to convert point cloud to a searchable surface = 1 minute, 58 seconds
Time taken to palpate the liver = 5 minutes, 27 seconds
Time taken to perform the entire surgical procedure = 10 minutes, 51 seconds
```

Figure 26: FVD Times

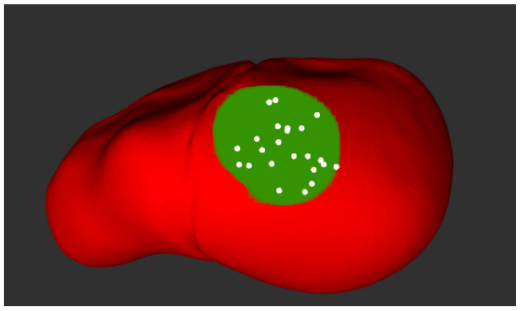
```
Time taken to 3D-scan the liver = 3 minutes, 36 seconds
Time taken to convert point cloud to a searchable surface = 2 minute, 58 seconds
Time taken to palpate the liver = 12 minutes, 47 seconds
Time taken to perform the entire surgical procedure = 19 minutes, 21 seconds
```

Figure 27: FVD Times with More Noise

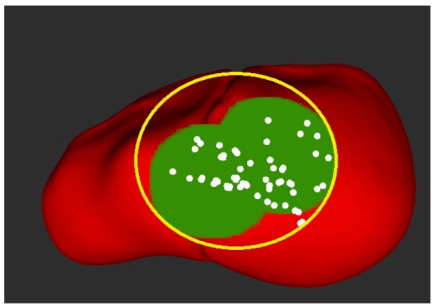
The final requirements that we measured during the FVD were M.P.9 and M.P.10, which relate to the accuracy of detection of cancerous tissue. We define tumor tissue in our system as the minimum bounding circle that encompasses all points we classify as a tumor during palpation. We define misclassified tissue as any tissue that is actually healthy that we decide is cancerous based on this minimum bounding circle. Through FVD and FVD encore, we tested various tumor numbers and shapes in order to ensure that our system worked as intended in general cases. The results are in Figure 28.



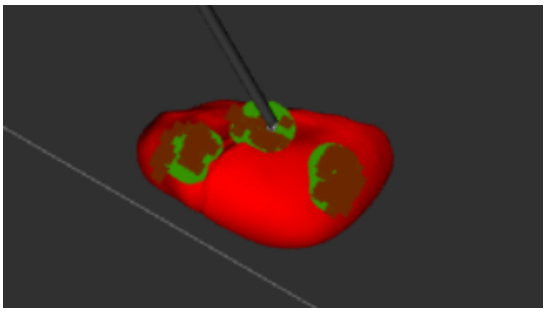
(a) 1 Tumor Results



(b) 1 Tumor With Noise



(c) 1 Oddly-Shaped Tumor



(d) 3 Tumors

Figure 28: Results with Different Tumor Shapes / Number

Figure 28a, the single tumor was detected with 100% cancerous tissue detection and with 1.69% healthy misclassified tissue. In Figure 28b, the single tumor with more noise added to the force sensor was detected with 100% cancerous tissue detection and with 2.49% healthy misclassified tissue. In Figure 28c, the single tumor that was oddly shaped was detected with 100% cancerous tissue detection and with 9.98% healthy misclassified tissue. Finally, in Figure 28d, the multiple tumors were each detected with 100% cancerous tissue detection and with 10.89% healthy misclassified tissue in total. The first two scenarios here meet our desired performance requirements. The second scenario with an oddly-shaped tumor meets our performance requirements. With multiple tumors, we meet our requirement for catching all the cancerous tissue and miss out on the misclassification error by 0.89%.

8.8 Strong and Weak Points

8.8.1 Strengths

Our robotic surgical system can autonomously create a 3D point cloud of the liver and palpate the region of interest to localize embedded tumors with an accuracy of 100% nearly all the time. Our system can perform this procedure on any organ deemed safe for robotic-assisted surgery and can localize tumors of arbitrary shapes. Most of the components that we have designed in simulation perform well in the target surgical environment individually and collectively. Each component that we designed performs within the time limits, which is crucial to being useful in a real surgery. The Blaser can generate accurate line scans that, in turn, allow our point clouds to be generated very well by implementing a simple scanning algorithm. Although we have not had to segment the liver out in the simulation, our segmentation algorithm performs well under multiple lighting conditions on hardware, which could guide the rest of the procedures following. Our motion estimation algorithm can estimate the frequency of the liver motion within 0.05 Hz of the ground truth, implying that while drift in the phase of the liver's position would accumulate over time and require correction, it would not cause a drastic offset. The Gaussian Processes-based palpation of our system balances exploration and exploitation to localize embedded tumors in the assigned period. Additionally, our motion simulation platform can replicate many different types of movements and holds the phantom liver and stomach well to simulate the motion of the organs realistically.

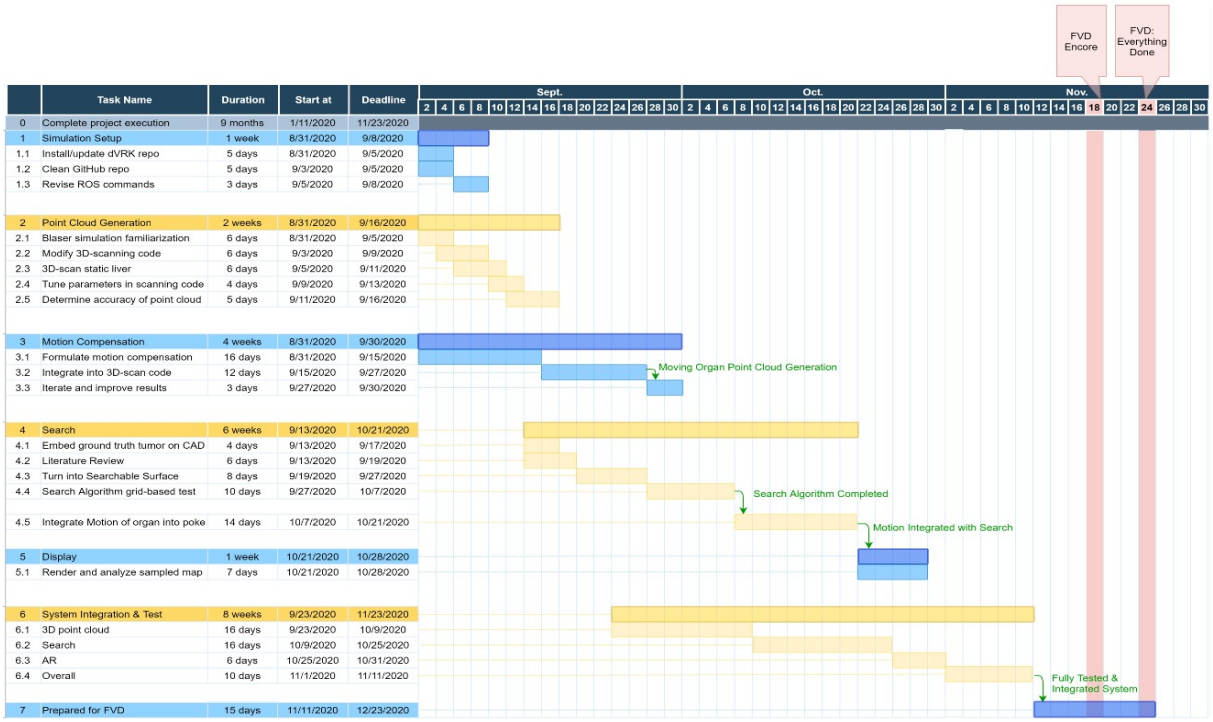
8.8.2 Weaknesses

Since we did not develop a stereo camera in simulation, we were not able to make use of a state estimation algorithm such as the Extended Kalman Filter(EKF) to correct the drift in the phase of the liver's position due to the difference between the ground truth and the estimated frequency of the liver's motion. Our system compensates for the ground truth liver frequency, but in the real use case, the system will have to estimate the frequency within a certain margin of error and correct the drift that accumulates over time due to this error. Another weakness of our system is the metric we use to quantify the accuracy of tumor localization- the minimum bounding circle. Once the robot has completed the palpation procedure, our algorithm draws a minimum bounding circle around each cluster of tumor locations. While this ensures that 100% of the tumor(s) is correctly localized, more tissue will be misclassified as cancerous if the tumor is not perfectly circular. The percentage of healthy tissue misclassified scales with the number of embedded tumors, which means a significant portion of healthy tissue would have to be resected when there are multiple tumors. Thirdly, the Gaussian Processes-based palpation we employ has the robot arm swing from one portion of the liver to another as it palpates the organ. In the case where the liver has 2 or 3 embedded tumors, the localization of these happens parallelly, causing the palpation to take a minute or two longer than if the robot tried to palpate around each suspected tumor completely before moving on to the next.

9 Project Management

9.1 Schedule

Figure 29 shows a detailed Gantt Chart that granularizes the schedule we had made in Spring 2020 and managed to follow, more or less, through Fall 2020.



	Task Name	Duration	Start at	Deadline
0	Complete project execution	9 months	1/11/2020	11/23/2020
1	Simulation Setup	1 week	8/31/2020	9/8/2020
1.1	Install/update dVRK repo	5 days	8/31/2020	9/5/2020
1.2	Clean GitHub repo	5 days	9/3/2020	9/5/2020
1.3	Revise ROS commands	3 days	9/5/2020	9/8/2020
2	Point Cloud Generation	2 weeks	8/31/2020	9/16/2020
2.1	Blaser simulation familiarization	6 days	8/31/2020	9/5/2020
2.2	Modify 3D-scanning code	6 days	9/3/2020	9/9/2020
2.3	3D-scan static liver	6 days	9/5/2020	9/11/2020
2.4	Tune parameters in scanning code	4 days	9/9/2020	9/13/2020
2.5	Determine accuracy of point cloud	5 days	9/11/2020	9/16/2020
3	Motion Compensation	4 weeks	8/31/2020	9/30/2020
3.1	Formulate motion compensation	16 days	8/31/2020	9/15/2020
3.2	Integrate into 3D-scan code	12 days	9/15/2020	9/27/2020
3.3	Iterate and improve results	3 days	9/27/2020	9/30/2020
4	Search	6 weeks	9/13/2020	10/21/2020
4.1	Embed ground truth tumor on CAD	4 days	9/13/2020	9/17/2020
4.2	Literature Review	6 days	9/13/2020	9/19/2020
4.3	Turn into Searchable Surface	8 days	9/19/2020	9/27/2020
4.4	Search Algorithm grid-based test	10 days	9/27/2020	10/7/2020
4.5	Integrate Motion of organ into poke	14 days	10/7/2020	10/21/2020
5	Display	1 week	10/21/2020	10/28/2020
5.1	Render and analyze sampled map	7 days	10/21/2020	10/28/2020
6	System Integration & Test	8 weeks	9/23/2020	11/23/2020
6.1	3D point cloud	16 days	9/23/2020	10/9/2020
6.2	Search	16 days	10/9/2020	10/25/2020
6.3	AR	6 days	10/25/2020	10/31/2020
6.4	Overall	10 days	11/1/2020	11/11/2020
7	Prepared for FVD	15 days	11/11/2020	12/23/2020

Figure 29: Gantt Chart depicting Fall Task Breakdown

Our biggest challenge in August 2020 was having to set up a simulation environment for the surgical system and making changes to the scope of the project. Our mentor, Nico Zevallos, at the Biorobotics lab helped us make a head start by creating a simulation for the Blaser sensor that we could then import into Rviz. We spent the first two weeks of the semester organizing our GitHub repository and setting up the simulation infrastructure. Keeping in mind that we had about 10 weeks until FVD, we realized it was important for us to be clear on exactly what we could accomplish and direct our efforts towards that. With the Blaser simulation in place, we revised the palpation module: instead of developing a force sensor in simulation, we decided to arbitrarily assign tumors on the liver's surface and make the palpation process a sampling-based search on the liver's surface. Additionally, we decided not to develop a stereo camera in the simulation since we had successfully demonstrated the camera-robot registration procedure on hardware in Spring 2020 and because the modified scope of our project did not require a stereo camera to complete the surgical procedure.

Once we had some clarity on the revised scope and requirements of our system, we found it relatively straightforward to follow through with it. In September, we realized that the Blaser simulator was laggy and worked slightly differently than the one on hardware that we had experience with; we fixed these issues by making changes to the config file. The script we used to guide the robot to scan the liver remained unchanged

but we spent nearly three weeks in October on point cloud processing, i.e., converting the dense point cloud obtained from the Blaser into a trajectory that the robot could follow to palpate the liver. A major chunk of October was also spent in devising a motion compensation scheme to allow the robot to scan and palpate the moving liver with no relative motion between them. The last couple of weeks leading up to the FVD were dedicated to implementing both the Gaussian Process-based palpation and the uninformed palpation to determine which of the two gave us better results while also satisfying our performance requirements. Although we made it a point to make integration of subsystems a continual process, we dedicated a full week leading up to the FVD to write a script that could perform the entire surgical process autonomously.

We believe our scheduling was effective for a couple of reasons. Firstly, the experience of working on the project in Spring 2020 made us aware that technical progress usually ends up taking a little longer than we account for, so this time around, we decided to add a few buffer days between tasks to allow for an extension. Secondly, we have had to switch from hardware to simulation, and anticipating initial hiccups, we revised the scope of our project to only include components we knew we could complete within the given frame of time. This proved to be a sensible decision also because the dates of the FVD and the FVD Encore were advanced by two weeks. Finally, we made sure to stay organized and communicative as a team, ready to help each other out in making sure tasks were completed before the deadline.

9.2 Parts List and Budget

Table 9: Bill of Materials

What	Purpose	Price	#	Spare	Total
3D Anatomy Cad mode	Organ Model	$170	1	0	$170
Dynamixel XL-320 servo motors	Platform	$21.90	6	2	$175.20
OpenCM9.04-C Micro-controller	Platform	$19.90	1	1	$39.80
USB Downloader (LN-101)	Platform	$14.90	1	1	$29.80
1/4 in Acrylic and Delrin Sheets	Platform	$20	2	0	$40
Robotis Rivet Set (RS-10)	Platform	$5.50	1	0	$5.50
Nylon Ball Joint M3 x 24mm x 3mm	Platform	$1.82	6	2	$14.56
Smooth On Ecoflex 00-10	Organ Model	$41.99	1	0	$41.99
Smooth-On Universal Mold Release	Organ Model	$417.37	1	0	$17.37
Silc-Pig Silicone Pigment	Organ Model	$27.10	1	0	$27.10
Dynamixel XL-320 servo motors	Organ Model	$21.99	1	0	$41.99
Smooth On Ecoflex 00-10	Organ Model	$41.99	1	0	$41.99
Micro USB Cable 10ft	Organ Model	$9.99	1	0	$9.99
				Total Spent:	$700.90

The total budget for the project is $5,000 and we spent a total of $700.90 which is 14.02% of the total budget. The moving platform is the most significant expense in our system. Table 9 lists these parts along with their costs to form our bill of materials. Since our project was switched to simulation, we have not had to buy any components in Fall 2020, leaving most of our budget unused. We used everything that we anticipated to use notwithstanding the spare parts and were still significantly under our total budget, which we consider a success.

9.3 Risk Management

Table 10: Risk Description and Mitigation
L: Likelihood
C: Consequence
Req.: Requirement Affected

#	Risk	Type	Mitigation	L	C	Req.
R1	Integration issues between subsystems	Technical/ Schedule	Integrate subsystems continually and test each subsystem before integrating	5	5	All
R2	Inability to access robot and sensors in the lab due to COVID-19	Technical/ Schedule	Continue the development of the system in a customized simulation environment	5	5	All
R3	Low robustness of tracking	Technical	Develop a simple prediction model based on past data	3	2	M.P.6
R4	Complexity in turning 3D point cloud to searchable surface	Technical	Make a head start on this task or opt for a simple & efficient geometric model	3	5	M.P.7
R5	Fail to palpate liver due to challenges in using force sensor	Technical	Do palpation in simulation based on CAD model of liver and simulated force readings	4	4	M.P.7 M.P.8
R6	Fail to implement motion compensation in scanning code	Technical	Revert to static organ assumption	4	5	M.P.5
R7	Fail to implement motion compensation in palpation code	Technical	Revert to static organ assumption	4	5	M.P.7 M.P.8
R8	PSM2 of dVRK cannot be fixed in the Fall	Technical/ Schedule	Use PSM1 for both scanning and palpation; detach Blaser from PSM1 after scanning liver and mount force sensor on the same arm to palpate liver	4	2	M.P.9
R9	Difficulty in registering RealSense & robot coordinate frames	Technical	Continue to use the Lab's stereo camera but modify liver motion tracking pipeline	3	4	M.P.4 M.P.6
R10	Malfunctioning of Blaser sensor	Technical	If attempts to fix the Blaser fail, buy a miniature off-the-shelf laser scanner	2	5	M.P.3 M.P.5
R11	MSP breaks / servo motors that actuate MSP are damaged	Technical	Handle MSP with care & buy servo motor spares for emergencies	2	3	M.P.6
R12	Search and Palpation development postponed due to Sensor Simulation Development	Schedule	Develop the simulation during summer break	2	3	M.P.7 M.P.8

In Spring 2020, while deciding our Fall 2020 Schedule in light of the worsening COVID-19 pandemic, we identified several risks. The two most critical were that the Patient Side Manipulator 2 (PSM2) of the dVRK did not work and that the COVID-19 pandemic could restrict access to the Biorobotics Lab. Table 10 shows our management strategy for all the risks we had identified, and a corresponding likelihood-consequence table is shown in Figure 30.

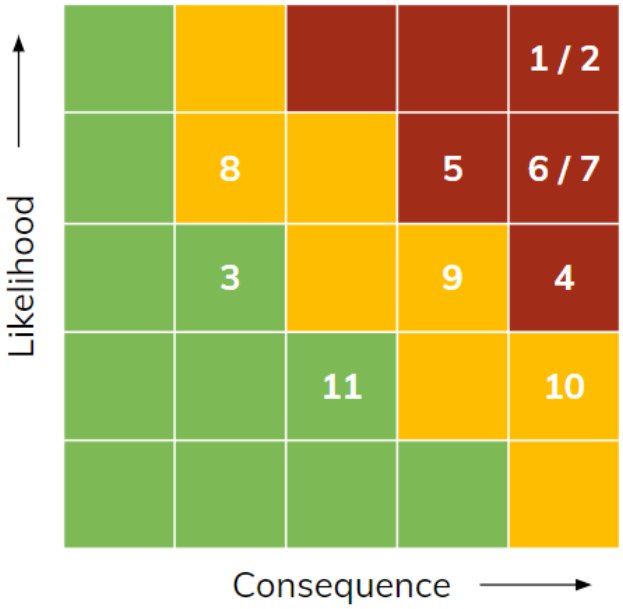


Figure 30: Risk Likelihood-Consequence Table

We had planned to go to the Biorobotics Lab over the Summer and fix the robot with the help of our mentor. However, the Campus was closed through the Summer and we were not able to go back in and fix the robot arm. Halfway through the Summer, we realized that COVID-19 would not pass anytime soon and were advised by our sponsor to switch our project to simulation so that we would be less affected by the pandemic's progress. While staying hopeful of being able to go back to Campus, we began setting up a simulation environment to complete the rest of our project, averting risks R2, R8, and R12. We consider our foresight to be one of the biggest strengths of our risk mitigation strategy. Our mentor at the Biorobotics Lab helped us get the Blaser simulator up and running by the end of June 2020. Since we were all busy with our internships over the Summer, we could only meet every fortnight or so. We spent a few weeks discussing what components of our original architecture to retain in the simulation environment. Our frequent meetings with our sponsor and our mentor proved to be an enormous helping hand in enforcing our risk mitigation strategy.

By mid-August, we were quite sure that our project would have to be completed in simulation and that a force sensor would not be part of it, averting risks R5, R9, R10, and R11. Formulating a motion compensation scheme took up nearly all of October 2020 but once we succeeded in developing a working prototype, risks R6 and R7 could be put aside. Once we had a motion compensation scheme and a point cloud filtering algorithm in place, risks R3 and R4 were averted. Risk R1 concerning integration issues between subsystems was overcome by making it a continual effort that we dedicated sufficient time to. Overall, we believe our risk management process was successful because we anticipated critical risks and worked on mitigating them over the Summer. This bought us valuable time to make progress towards our performance requirements in the Fall.

10 Conclusion

10.1 Lessons Learned

Developing the Chopsticks Surgical System was a challenging but extremely gratifying endeavor. Not only did each of us get the opportunity to hone our technical skills but also develop collaborative skills that will benefit us in the long run. Our key takeaways are elaborated below:

- **Be organized:** Over time, we came to realize that we function much better when we follow a methodical approach to the task at hand. This applies to writing and maintaining clean code, storing documents in the appropriate folder on Google Drive, not procrastinating on tasks to be done, and making each person accountable for the job entrusted to them.
- **Write robust and scalable code:** Although each of us worked on separate branches on GitHub up until a few weeks before the FVD, we would often have to test each other's codes out. It became apparent to us very soon that we could not use absolute paths to real files as inputs since that would only work on the system of who wrote the script. Hence, we began to use relative paths and passed in user-defined parameters such as liver amplitude and frequency as arguments. This ensured we did not have to make modifications to several files when running a teammate's script.
- **Test the system frequently:** Software is a tricky business and it is often hard to predict the ramifications of a seemingly small change on the pipeline of the project. We made it a point to test the system rigorously and frequently, and this helped us catch bugs early on.
- **Keep it realistic:** When we had to switch from hardware to simulation, we realized that while it is great to be ambitious and push our limits, we had to be realistic and define the scope of the project to be something we knew we could get done in 10 weeks' time. This was particularly important because we had assignments and projects to do in other courses, and the hunt for a full-time job to do on the side.
- **Communicate often and effectively:** In Spring 2020, one of the biggest challenges we faced was communicating effectively as a team. Each person would work on his or her tasks but never take the time out to update the others on the progress made and potential risks to the system. This would lead to unnecessary confusion and stress before progress reviews. Our meetings had no fixed schedule and were on an ad hoc basis. In Fall 2020, we laid out a fixed schedule for meetings and made sure to update the team on our individual progress. This helped us get things done quicker and with more ease.

10.2 Future Work

Due to the pandemic, the da Vinci Research Kit at the Biorobotics Lab is not freely accessible. Therefore, we have had to finish our project in simulation. Currently, we use the ground truth motion parameters in the motion compensation module of the

controller. If we were to use the estimated parameters instead, the cumulative drift would eventually cause the robot arm to collide with the liver. A logical extension of our work is to incorporate images from the camera feed to correct the drift that arises due to the error in motion estimation. Moreover, our tumor modeling in simulation does not reflect the actual physics of tumor localization. To perform surgery on hardware, we would have to mount a miniature force sensor on the robot arm and program the robot to palpate the organ with motion compensation. We made the assumption that embedded tumors are situated close enough to the surface of the organ to be detected by the force sensor though, in reality, tumors could be embedded quite deep in the organ and would require the surgeon to cut into the organ to fully localize it. We had also assumed that the organ does not deform as a result of palpation, but studies show that not all organs are perfectly elastic. We need to investigate how to account for organ inelasticity in our palpation module. Finally, although we performed the entire surgical procedure using just one arm of the dVRK, it would be far more efficient and effective to make full use of the dual-arm robot that the dVRK is. The Blaser would be mounted on PSM1 and the force sensor on PSM2, coordinating and working together like a pair of chopsticks.

References

[1] N. Zevallos et al., *"A surgical system for automatic registration, stiffness mapping and dynamic image overlay,"* 2018 International Symposium on Medical Robotics (ISMR), Atlanta, GA, 2018, pp. 1-6.

[2] Rangaprasad, Arun Srivatsan & Wang, Long & Ayvali, Elif & Simaan, Nabil & Choset, Howie. (2016). *Simultaneous Registration and Stiffness mapping of a Flexible Environment using Stiffness and Geometric Prior.*

[3] H. Salman et al.,*"Trajectory-Optimized Sensing for Active Search of Tissue Abnormalities in Robotic Surgery,"* 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 5356-5363.

[4] L. Li, B. Yu, C. Yang, P. Vagdargi, R. A. Srivatsan and H. Choset, *"Development of an inexpensive tri-axial force sensor for minimally invasive surgery,"* 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 906-913.

[5] Patel, Vatsal & Krishnan, Sanjay & Goncalves, Aimee & Goldberg, Kenneth. (2017). *SPRK: A Low-Cost Stewart Platform For Motion Study In Surgical Robotics.*

[6] Bjorne Stemkens et.al, *"Image-driven, model-based 3D abdominal motion estimation for MR-guided radiotherapy"* in Institute of Physics and Engineering in Medicine, 2016 Phys. Med. Biol. 61 5335.

[7] M. Bowthorpe and M. Tavakoli, *"Physiological Organ Motion Prediction and Compensation Based on Multirate, Delayed, and Unregistered Measurements in Robot-Assisted Surgery and Therapy,"* in IEEE/ASME Transactions on Mechatronics, vol. 21, no. 2, pp. 900-911, April 2016.

[8] E. Ayvali, R. A. Srivatsan, L. Wang, R. Roy, N. Simaan and H. Choset, *"Using Bayesian optimization to guide probing of a flexible environment for simultaneous registration and stiffness mapping,"* 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 931-936.

[9] D. E. Whitney, *"Resolved Motion Rate Control of Manipulators and Human Prostheses,"* in IEEE Transactions on Man-Machine Systems, vol. 10, no. 2, pp. 47-53, June 1969, doi: 10.1109/TMMS.1969.299896.