

Camera-based Augmented Reality Endoscope Auxiliary System

By

Boyang Zhou

Jianhan Ma

Tengyue Wang

Zhenyu Zong

Final Report for ECE 445, Senior Design, Spring 2021

Instructors: Mark Butala, Arne Fliflet and Timothy Lee

TAs: Shu Xian Lai, Sophie Liu and Xinyi Xu

21 May 2021

Project No. 9

Abstract

In modern surgery, doctors inspect the inner organs with the endoscope to minimize the surgical wound. Medical vision techniques like X-ray or magnetic resonance imaging can help doctors find tumor location in patients' organs. However, doctors have to locate the tumor by his or her empirical skills as the endoscope cannot see through the surface of organs. Our project provides a solution to aid doctors in surgery to find the location of tumor inside organ. By aligning the Unity virtual scene containing the tumor and the real scene captured by the endoscope controlled by robot arm, we utilize the AR auxiliary system capable of tracking and tagging the tumor location and displaying the tag on the endoscope's image output. Our system will help the doctors locate the tumor inside the organ and reduce the risk of surgical accidents due to doctors' wrong judgment. This document will go into further detail regarding our solution of the AR auxiliary system.

Contents

1 Introduction	1
1.1 Problem Statement.....	1
1.2 Solution	1
1.3 High-Level Requirements.....	1
1.4 Physical Design.....	2
1.5 Block Diagram	2
2 Design.....	3
2.1 Power Supply	3
2.1.1 220 V AC.....	3
2.1.2 DVE Switching Adapter	3
2.2 Manipulator Module.....	3
2.2.1 OpenCR Board.....	3
2.2.2 RM-X52-TNM Robot Arm.....	4
2.3 Laparoscope Module	4
2.3.1 Connector	4
2.3.2 Endoscope.....	5
2.3.3 Liver Model	5
2.4 Control and Process Unit	6
2.5 Tolerance Analysis	8
3 Design Verification	10
3.1 Power Supply	10
3.2 Manipulator Module.....	10
3.3 Laparoscope Module	11
3.4 Control and Process Unit	11
4 Costs.....	14
5 Conclusion.....	15
5.1 Accomplishments.....	15
5.2 Uncertainties.....	15
5.3 Ethical considerations.....	15

5.4 Future work.....	16
References	17
Appendix A ROS Topic and Message Table for Robot Arm	18
Appendix B Requirement and Verification Table	18

1 Introduction

1.1 Problem Statement

Modern medical doctors inspect the inner organs using the endoscope for minimally invasive surgery [1]. The minimally invasive surgery is different from the open operations that cause a larger wound [2]. Because of the smaller damage and shorter recovery time for the patients, endoscope operations have widespread applications [3]. Nowadays, although the location of the tumor in patients' body can be easily found by imaging techniques like X-ray or magnetic resonance imaging, in surgical procedure, the doctor will still have to find the focus by his or her empirical skills since the endoscope cannot see inside the tissue directly. If not appropriately taken, iatrogenic injury in the clinical operation may lead to serious complications and apparatus damage [4]. To avoid such a medical accident, it would be necessary to provide the focus to the doctor.

1.2 Solution

Our goal is to design an AR endoscope auxiliary system that is able to use the focus's location information in the 3D organ model, assumed to be known by preprocessing the X-ray or magnetic resonance image, to produce a clear tag of the focus on the 2D image captured by the endoscope. To be specified, our system concentrates on tagging the tumor position in the liver model. The liver model is a 3D-printed physical model, which imitates the shape of a liver. Instead of manual operation of the endoscope, our system uses ROBOTIS company's OpenMANIPULATOR-X robotic arm to control the endoscope on the operation table for stability and easy access to endoscope's pose information. The accessed pose information can be used to align the virtual camera in Unity virtual scene, which contains the tumor location derived by medical vision techniques with the real-world endoscope which captures the real-world organ surface information. As for the virtual camera, we use it to sense the tumor location in virtual scene and use OpenCV to capture the tumor and generate a tag for it. Our system applies overlapping to the tagged virtual image and the image of the endoscope. In this way, the AR can be realized by the aligned virtual and real scene overlapping so that doctors can know where to do the operation by looking at the tagged image on the screen directly.

1.3 High-Level Requirements

1. The endoscope returns images of the infected liver with a resolution of 1920 x 1080 and a frame rate of 30 fps to avoid unclear or dysfluent image flow affect the success of the surgery.
2. The endoscope has a data transmission latency of 3 s. The robot arm's velocity should be constrained to allow our system to have a latency error tolerance to avoid unexpected damage to patients. The project will constrain the moving speed of the endoscope within 0.0082 m/s.
3. Accurate and stable display of the tagged tumor can be overlaid on endoscope's images, with position error less than 0.5 cm and rotation error less than 1 degree. The tagged tumor can move according to the endoscope movement in the virtual image.

1.4 Physical Design

Figure 1 shows the workspace of the system. PC controls the movements of robot arm and process image data captured by endoscope. The overlaid 3D model will be displayed in Unity3D scene.

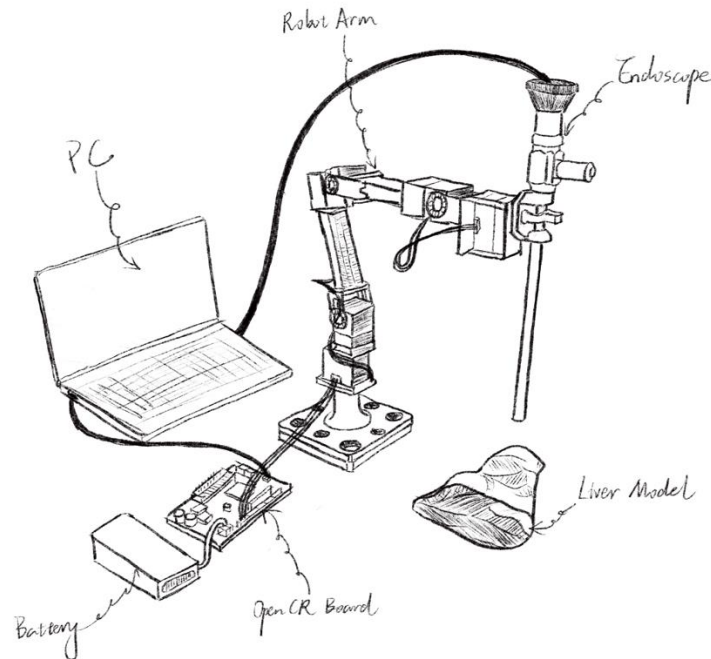


Figure 1. Physical design of the whole system

1.5 Block Diagram

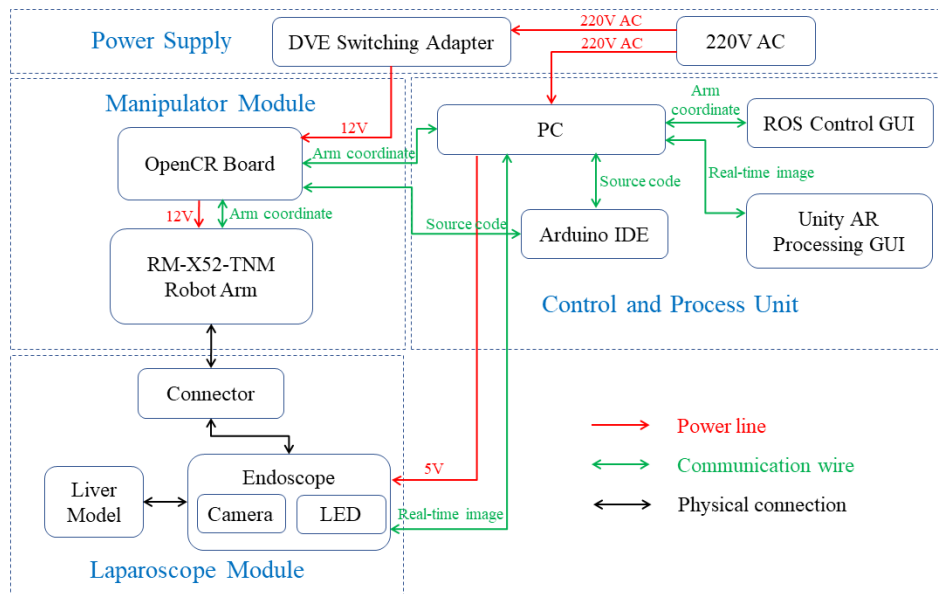


Figure 2. Block diagram including power supply, manipulator module, laparoscope module and control & process unit

2 Design

The system consists of four components: power supply, manipulator module, laparoscope module, and control & process unit. The power supply unit provides stable and safe 12 V power to the OpenCR board. The manipulator module contains the OpenCR board and robot arm. OpenCR is the communication interface between PC and robot arm. It also supplies 12 V power to the robot arm. This module ensures the communication Laparoscope module, consisting of a rigid shaft, endoscope and liver model, simulates the operation scenario and captures real-time 1920 x 1080 pixels 30 fps video data. Control & Process unit is the processing center to process data and sends control instructions. The overlaid scene will be displayed in Unity. The ROS control GUI will be used to control the velocity of the robot arm and achieve the velocity upper limit constrain of 0.0082 m/s. In the final integration step, PC will gather the robot arm coordinate information and the real-time image and use the code we designed to process the data to generate the endoscope real-time image with tumor position tagged, satisfying the requirement that position error less than 0.5 cm and rotation error less than 1 degree.

The detailed block design is shown in Figure 2. Next, we will introduce our design procedure and details of each block and all components.

2.1 Power Supply

All devices need their required power supply to work. The robot arm is powered by a switching adapter converting 220 V AC to 12 V 5 A DC. The power is supplied to OpenCR board's power port at first. The board will output the power to the robot arm through its TTL cable. For the convenience of demonstrating, we will use the battery to supply power.

2.1.1 220 V AC

220 V alternating current is the Chinese standard civil voltage. This is the power source of PC. The DVE Switching Adapter will convert the power to supply OpenCR board and robot arm.

2.1.2 DVE Switching Adapter

This is a power supply and battery charging device to power the robot. It converts 220 V AC to 12 V 5 A power to OpenCR board through power cable.

2.2 Manipulator Module

Manipulator module contains robot arm and OpenCR board. This module guarantees the communication between the control unit and the laparoscope module. It receives control instructions from control & process unit and implements moving instructions of robot arm.

2.2.1 OpenCR Board

The OpenCR 1.0 board is the communication interface between PC and robot arm. PC sends source code to OpenCR board through Arduino IDE so that the board can implement real-time computation to provide forward and inverse kinematics to robot arm. It receives 7 - 24 V power from DVE Switching Adapter through power cable and provides power to robot arm through TTL cable. TTL cable also implements data flow between PC and robot arm.

2.2.2 RM-X52-TNM Robot Arm

We choose OpenMANIPULATOR-X (RM-X52-TNM) robot arm from ROBOTIS company. This robot arm has five DOFs, including four DOFs for joints and one DOF for gripper. The gripper on the robot arm uses a 3D-printing mechanical connector to hold the endoscope and implement kinematics instructions to the endoscope. Because this gripper cannot hold the thin endoscope, we replace it with our designed connector in Figure 3.2.

OpenMANIPULATOR-X has the emergency stop functionality. When an emergency stop happens, the actuators sensing strong forces will stop immediately and cannot be turned on again. In this condition, we need to check the actuator's status before relaunching the robot arm again. This functionality can protect both the robot arm and patients.

The software control is based on ROS and OpenSource [5]. All five joint status are expressed in ROS messages and topics. The detailed message and topic structure can be seen in Appendix A. We will use the joint statuses in the message.

2.3 Laparoscope Module

The laparoscope module consists of a connector, endoscope device, and 3D printing physical liver model. It simulates as a prototype of actual abdominal surgery. The original design is that the manipulator is restricted by a fixed hole on the operation table, as shown in Figure 1. However, we find that actuators may get strong resistance force when the manipulator is moving. So, the operation table and the fixed pivot movement design are given up.

2.3.1 Connector

This is a 3D printing mechanical connector that aligns the endoscope with the robot arm's gripper, manufactured by our ME team member on the team. Figure 3.1 shows our original design, which replaces the fifth actuator with a connector to hold the endoscope. However, the robot arm will lose one DOF to rotate without the fifth actuator. The new design in Figure 3.2 solves that problem by using the robot arm's rotation journal bearing so that the endoscope can rotate in a plane with the range of -40 degrees to 40 degrees.

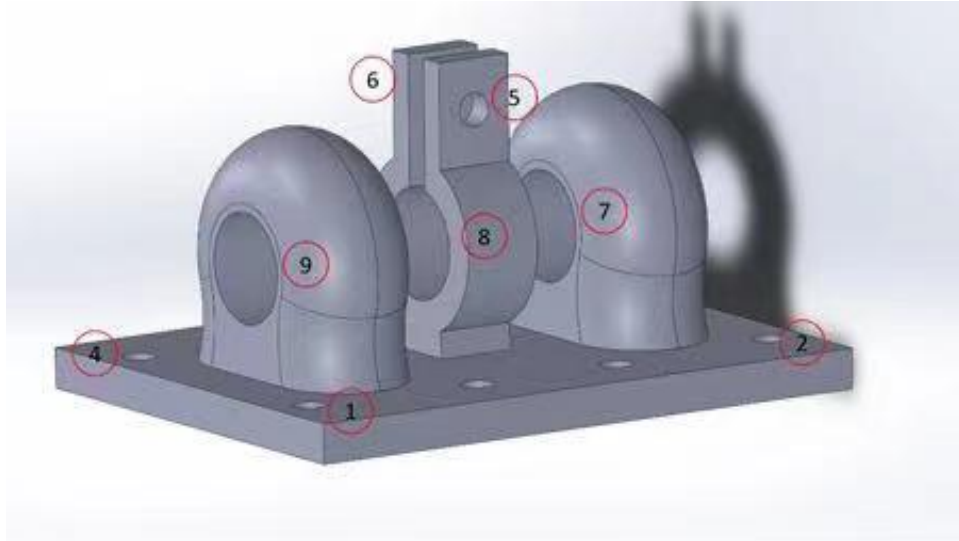


Figure 3.2. Connector replacing the fifth actuator

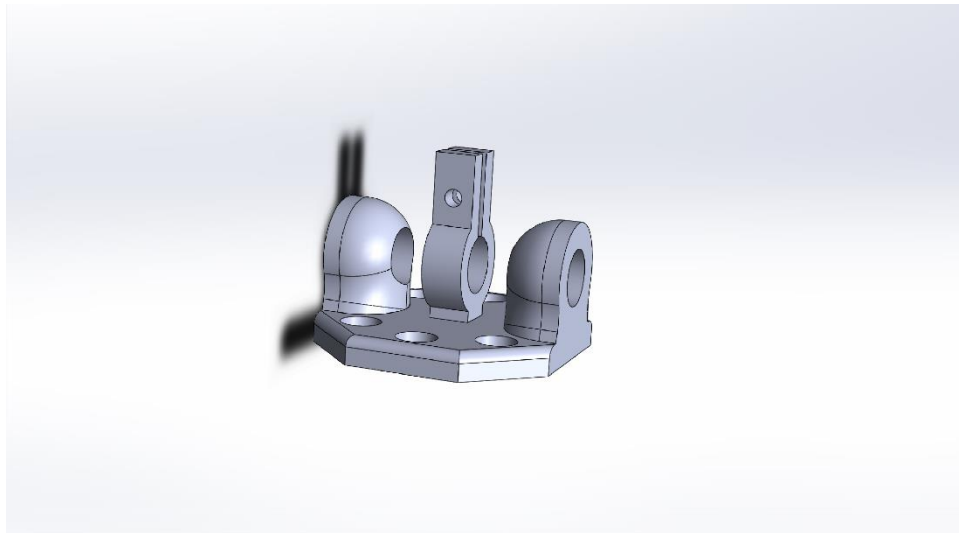


Figure 3.2 (continued). Connector aligning the endoscope with the fifth actuator

2.3.2 Endoscope

The endoscope consists of a camera and a LED. The LED lights up the abdominal environment to create a favorable condition for the camera. The camera captures images of the liver model with resolution of 1920 x 1080 pixels and frame rate of 30 fps. Those images are sent to PC for further processing in Unity. Both LED and camera get 5 ± 0.2 V power supply from PC.

2.3.3 Liver Model

We use the 3D printing liver model to simulate the patients' real liver. The model will be attached with focus labels to represent tumors. The liver model will fit the liver size of normal human beings, which is about 14 cm in diameter.

2.4 Control and Process Unit

The control & process unit is the software part of our system. It mainly has three parts: OpenCR board source codes, Unity C# scripts and Python scripts in Ubuntu virtual machine.

OpenCR board source code is the official scripts provided in Arduino IDE named 'usb_to_dxl'. After processing the program in Arduino IDE, we upload the codes to /dev/ttyACM0 port of OpenCR board. Note that sometimes the port cannot be accessed. That is because it lacks the right to be read and written. We should give it R/W authority using 'chmod' instruction. After that, the upload of the source code and 'jump_to_fw' implies uploading success.

Unity C# scripts have three parts: Rosbridge communication codes, image processing codes and camera calibration codes. Rosbridge provides Unity with JSON API to use ROS functionalities. It contains front ends to use WebSocket to communicate with Web Server and other nodes. In unity, we define the RosWebSocket using namespace named 'RosSocket'. This node can subscribe and publish message to other nodes using functions provided by Rosbridge. Another way to accomplish communication between unity and ROS is using WebGL. However, it needs to build the whole project as WebGL project to run on browser to use html scripts to transfer ROS information. We don't want our project to be built because the overlay images shall be displayed in the Unity workspace.

The image processing code is assembled in the C# script named *Imgcapture*. The goal of the script is to capture the location information of the tumor model in virtual scene and tags it on the endoscope image to create the AR effect. The script first gathers the visual information captured by the virtual camera in the Unity scene and the endoscope and store the information as two "Texture2D" type variables. The script calls a nuget package, named "opencvsharp", to accomplish the two texture's processing and overlapping. After gray scaling the virtual camera texture, a method in "OpenCVSharp" named "HoughCircles" was applied to find the location and radius of the sphere in the Unity scene which represents the tumor. With the location and radius, the script uses the "Circle" method in "OpenCVSharp" to draw a circle tag on the endoscope texture and returns an AR image containing the tumor's location information.

The camera calibration code is assembled in the C# script named "Forwardkinematics". The goal of the script is to receive the transform matrix sent by Ubuntu and use it to calibrate the virtual camera in the Unity scene and the endoscope held by the robot arm. The script calls the ROSBridge communication code described above and creates a template for it. Then it uses the template to read the transform matrix sent by the ROS in Ubuntu and stores it. The translational movement information is applied by the "transform.position" method to move the virtual camera to the target place, the rotation matrix is transformed into the Euler angle by the method named "LookAt" provided by Unity. And the orientation of the virtual camera is set by the "transform.eulerAngles" method with the Euler angle provided. After the transformation, the virtual camera and the endoscope will be aligned, and their visual information will also be aligned automatically.

In Ubuntu virtual machine, python scripts have two functionalities: Nodes to publish messages and forward kinematics calculation. In the listener_pub_py.py file, we define the node named '445_node' as

publish node in VM. This node subscribes all topics and message from robot arm. Then we use those messages to calculate forward kinematics at the end effector located at the endoscope's camera. Then we define a new message named 'forward_kinematics', which is 4 x 4 matrix containing the forward kinematics information. Its corresponding Euler angles are stored in the matrix's fourth row because the fourth row of the forward kinematics matrix usually is [0, 0, 0, 1], which does not contain any helpful information. The 'forward_kinematics' message is published to the node in Unity through WebSocket IP: 192.168.56.103: 9090. SPIN_RATE specifies the publishing rate with unit Hz. There are two methods to calculate forward kinematics: Denavit - Hartenberg parameters (D-H) and Product of Exponentials formula (PoE). We choose PoE method to do calculations because the screw and exponential matrix calculation are much more convenient in Python. So PoE will be easier and more efficient compared with D-H method.

ROS provides GUI to control OpenManipulator robot arm. We choose open_manipulator_control_gui and open_manipulator_teleop_keyboard packages. Figure 4.1 shows the details of the former control GUI. After clicking 'Timer Start' button, we can check joints states and set the pose of the robot arm using this GUI. Figure 4.2 shows all keyboard control instructions. This keyboard control package supports both movements in task space and joint angle settings.

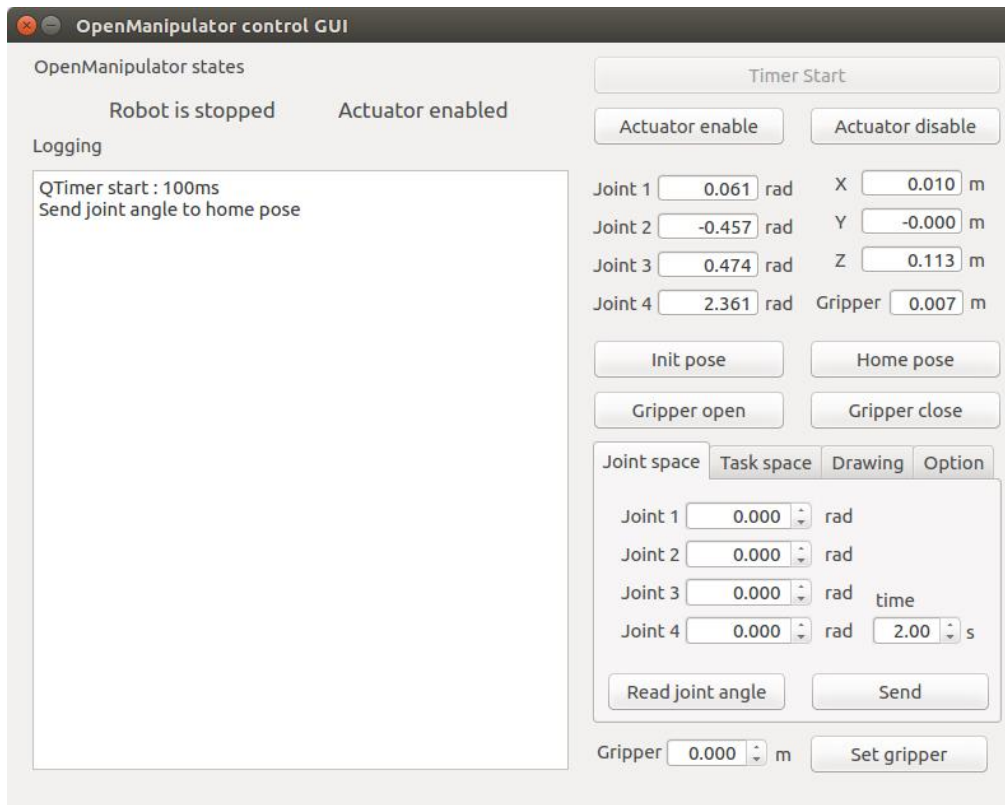


Figure 4.1. OpenManipulator control GUI

```

/home/zhenyu/catkin_ws/src/open_manipulator/open_manipulator_teleop/launch/open_
process[teleop_keyboard-1]: started with pid [8051]
[ INFO] [1618248709.725738005]: OpenManipulator teleoperation using keyboard sta
rt

-----
Control Your OpenManipulator!
-----
w : increase x axis in task space
s : decrease x axis in task space
a : increase y axis in task space
d : decrease y axis in task space
z : increase z axis in task space
x : decrease z axis in task space

y : increase joint 1 angle
h : decrease joint 1 angle
u : increase joint 2 angle
j : decrease joint 2 angle
i : increase joint 3 angle
k : decrease joint 3 angle
o : increase joint 4 angle
l : decrease joint 4 angle

g : gripper open
f : gripper close

1 : init pose
2 : home pose

q to quit
-----
Present Joint Angle J1: 0.000 J2: 0.000 J3: 0.000 J4: 0.000
Present Kinematics Position X: 0.000 Y: 0.000 Z: 0.000
-----

```

Figure 4.2. Keyboard control in terminal

2.5 Tolerance Analysis

A vital trade-off of our project should make is about the speed of the end effect of endoscope. For the manually operational endoscope, doctors who control the endoscope will handle the speed of the end effect dynamically to avoid doing damage to patients' organs. Since our project replace doctors' hands by the robot arm to control the movement of endoscope, we must find out a proper speed range of the robot arm's gripper to ensure the security.

Our endoscope has a data transmission latency of around 0.8 seconds. A proper endoscope velocity should allow the robot arm to handle the position error caused by the latency. To ensure the security, we make the worst-case assumption in which the data transmission latency is 1 second. Since liver tissue is very delicate, we set 0.005 m as the maximum deformation allowed for surgical security. Under the assumption, we can calculate the maximum velocity by the following equation:

$$V_{end-effect} = \frac{d_{Deformation}}{t_{Latency}} = \frac{0.00500}{1.00} = 0.00500 \text{ m/s} \quad (1)$$

Our endoscope is a rigid rod whose DOF is constrained by the hole in the operational table. In other words, we can view the intersection point of the endoscope and the operating table as a fixed point in the plane. Our robot arm uses a gripper to hold the endoscope and the gripper is always perpendicular to the endoscope.

By rigid body's properties we can easily get the angular velocity relation between the endoscope's end effect and the gripper of robot arm:

$$\omega_{end-effect} = \omega_{gripper} \quad (2)$$

Using the rigid body kinematics, the angular velocity can be transformed to the velocity:

$$V = \omega \times d \quad (3)$$

Our endoscope has a total length of 26.645 cm and for the sake of effective robot arm control the maximum depth it can reach into the abdomen is 19.253 cm. The gripper's point holds the endoscope has a distance of 4.512 cm from the intersection hole in the operating table. Therefore, to find the maximum velocity of the gripper, we only need to consider the maximum depth condition.

$$\omega = \frac{V_{end-effect}}{d_{max-depth}} = \frac{0.00500}{0.193} = 0.0260 \text{ rad/s} \quad (4)$$

$$V_{gripper} = \omega \times d = 0.0260 \times 0.0451 = 0.00117 \text{ m/s} \quad (5)$$

From the calculation above, we derive the absolute safe velocity for our gripper 0.00117 m/s. However, for the practical use of the endoscope, doctors should effectively get the image information through the endoscope. The absolute safe velocity is inefficient in switching the vision and even can lead to failure of surgery. Therefore, we make a trade-off between security and efficiency. We choose to calculate the proper velocity of the gripper using the typical depth of the endoscope for abdomen surgery which is 9 cm. The distance from the gripper and the intersection hole on the operating table can be calculated by:

$$d_{common} = d_{max-depth} - d_{common-depth} + L = 0.193 - 0.0900 + 0.0451 = 0.148 \text{ m} \quad (6)$$

And the equation below can give us the new maximum velocity of the gripper:

$$\omega = \frac{V_{end-effect}}{d_{max-depth}} = \frac{0.00500}{0.0900} = 0.0556 \text{ rad/s} \quad (7)$$

$$V_{gripper} = \omega \times d = 0.0556 \times 0.148 = 0.00821 \text{ m/s} \quad (8)$$

The proper velocity upper limit of the robot arm gripper balancing the security and efficiency is 0.00821 m/s.

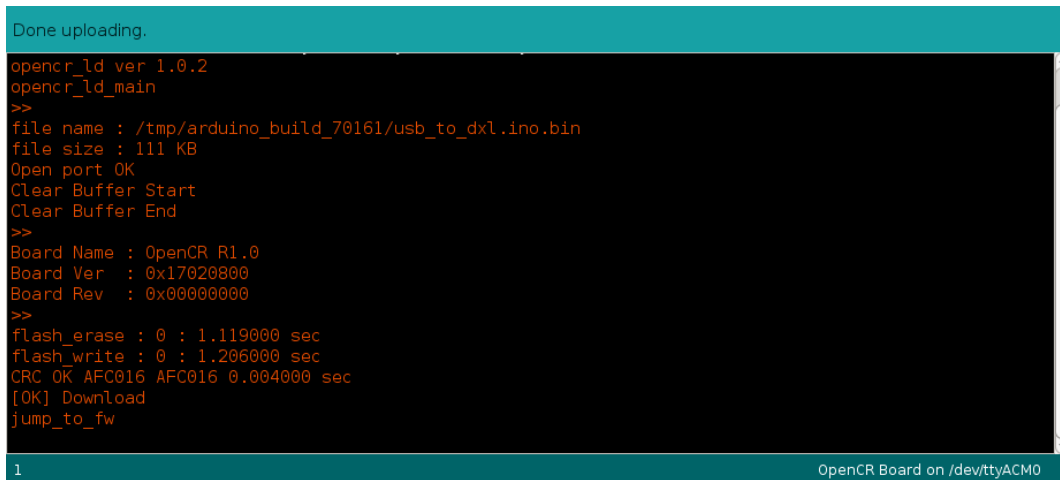
3 Design Verification

3.1 Power Supply

All power supply's input and output voltage are verified to satisfy their requirements. The detailed requirements and verification can be seen in Appendix B.

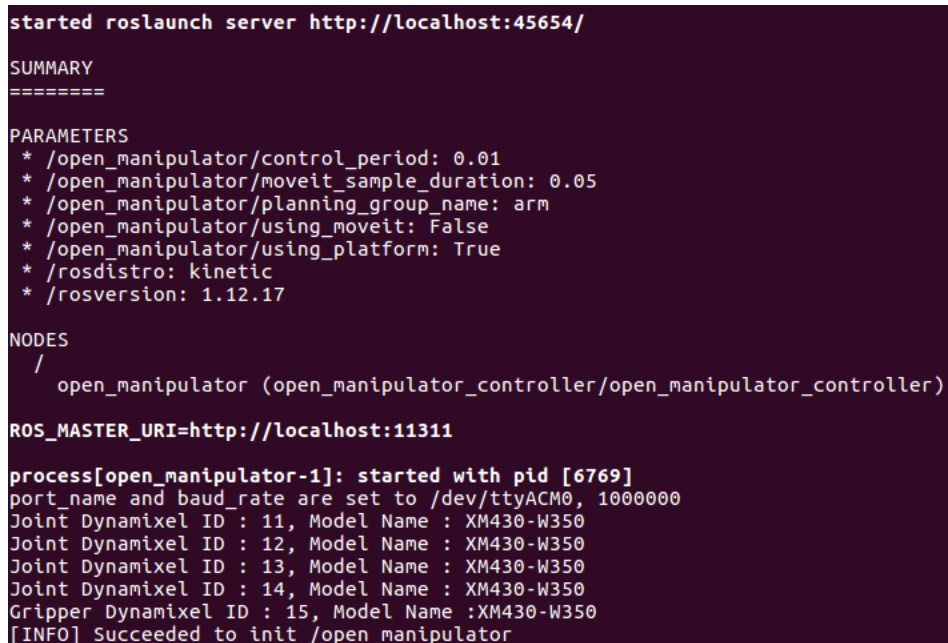
3.2 Manipulator Module

In Figure 5.1, 'jump_to_fw' implies that uploading source code to OpenCR board ttyACM0 port is prosperous. The robot arm can hold the endoscope with our designed connector in Figure 3.2, with slight robot arm deformation. The Roslaunch procedure is successful, as shown in Figure 5.2.



```
Done uploading.
opencr_ld ver 1.0.2
opencr_ld_main
>>
file name : /tmp/arduino_build_70161/usb_to_dxl.ino.bin
file size : 111 KB
Open port OK
Clear Buffer Start
Clear Buffer End
>>
Board Name : OpenCR R1.0
Board Ver : 0x17020800
Board Rev : 0x00000000
>>
flash_erase : 0 : 1.119000 sec
flash_write : 0 : 1.206000 sec
CRC OK AFC016 AFC016 0.004000 sec
[OK] Download
jump_to_fw
1 OpenCR Board on /dev/ttyACM0
```

Figure 5.1. Uploading outcomes in Arduino IDE



```
started roslaunch server http://localhost:45654/
SUMMARY
=====
PARAMETERS
* /open_manipulator/control_period: 0.01
* /open_manipulator/moveit_sample_duration: 0.05
* /open_manipulator/planning_group_name: arm
* /open_manipulator/using_moveit: False
* /open_manipulator/using_platform: True
* /roscdistro: kinetic
* /rosversion: 1.12.17
NODES
/
  open_manipulator (open_manipulator_controller/open_manipulator_controller)
ROS_MASTER_URI=http://localhost:11311
process[open_manipulator-1]: started with pid [6769]
port_name and baud_rate are set to /dev/ttyACM0, 1000000
Joint Dynamixel ID : 11, Model Name : XM430-W350
Joint Dynamixel ID : 12, Model Name : XM430-W350
Joint Dynamixel ID : 13, Model Name : XM430-W350
Joint Dynamixel ID : 14, Model Name : XM430-W350
Gripper Dynamixel ID : 15, Model Name : XM430-W350
[INFO] Succeeded to init /open_manipulator
```

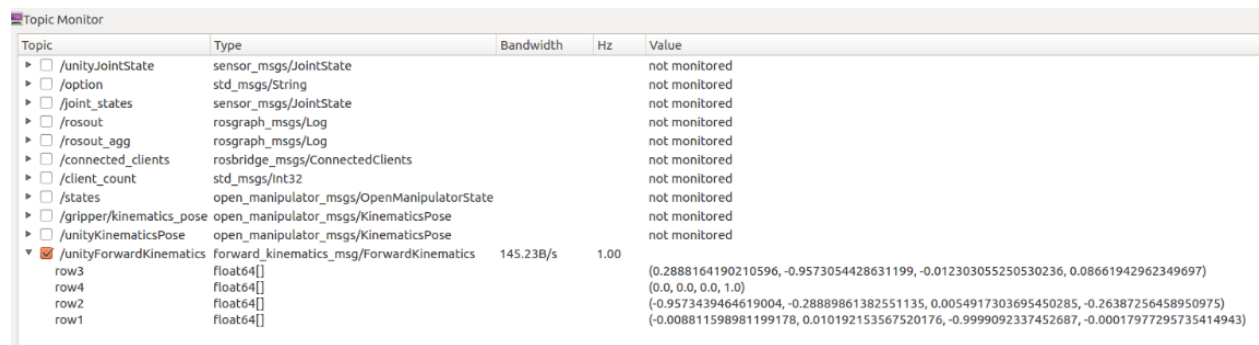
Figure 5.2. Roslaunch robot arm outcomes

3.3 Laparoscope Module

The connector can hold the endoscope tightly. The camera in the endoscope can capture images with 30 fps and the displacement is within the tolerance. The detailed requirements and verification procedures can be seen in Appendix B.

3.4 Control and Process Unit

The ROS communication program works well. In Figure 6.1, the rqt provides a topic monitor to display all ROS messages and topics published and subscribed by Virtual Machine Node. By ticking the check box, we can check the detailed information inside messages. The sending rate is controlled by the SPIN_RATE set in our python scripts. We choose the value of 1 Hz so that the program can have enough time to do forward kinematics calculation and avoid the clash with other processes.



Topic	Type	Bandwidth	Hz	Value
<input type="checkbox"/> /unityJointState	sensor_msgs/JointState			not monitored
<input type="checkbox"/> /option	std_msgs/String			not monitored
<input type="checkbox"/> /joint_states	sensor_msgs/JointState			not monitored
<input type="checkbox"/> /rosout	rosgraph_msgs/Log			not monitored
<input type="checkbox"/> /rosout_agg	rosgraph_msgs/Log			not monitored
<input type="checkbox"/> /connected_clients	rosbridge_msgs/ConnectedClients			not monitored
<input type="checkbox"/> /client_count	std_msgs/Int32			not monitored
<input type="checkbox"/> /states	open_manipulator_msgs/OpenManipulatorState			not monitored
<input type="checkbox"/> /gripper/kinematics_pose	open_manipulator_msgs/KinematicsPose			not monitored
<input type="checkbox"/> /unityKinematicsPose	open_manipulator_msgs/KinematicsPose			not monitored
<input checked="" type="checkbox"/> /unityForwardKinematics	forward_kinematics_msg/ForwardKinematics	145.23B/s	1.00	(0.2888164190210596, -0.9573054428631199, -0.012303055250530236, 0.08661942962349697) (0.0, 0.0, 0.0, 1.0) (-0.9573439464619004, -0.28889861382551135, 0.0054917303695450285, -0.26387256458950975) (-0.008811598981199178, 0.010192153567520176, -0.9999092337452687, -0.00017977295735414943)
row3	float64[]			
row4	float64[]			
row2	float64[]			
row1	float64[]			

Figure 6.1. rqt topic monitor

ROS controlling can also be accomplished through ROS GUI and keyboards. Figure 4.1 displays the detailed interface. After pressing “Timer Start” to launch the robot arm, we can give instructions to rotate any joints of the robot arm. Besides, our GUI also support the robot arm to move in task space x-y-z, as long as the destination is in the space where inverse kinematics can be calculated.

Figure 6.2 shows the Unity scene. There are two cameras, one green sphere and one white plane in it. The main camera is the virtual camera we want to calibrate with the real endoscope. We use the main camera to capture the information of the green sphere and overlap the green sphere virtual image with the real endoscope image to create AR effect. The green sphere is the representation of the tumor in our virtual scene and its location is designed to be matched with the location information of real tumor, which is already known by using the X-ray or magnetic resonance imaging. Another video camera is used to show the output of our system—the AR endoscope image in the game window of Unity 3D. It captures the video information on the white plane and displays it in the game window. The plane is used to receive the AR image output of our system and make it visible by the video camera in the Unity 3D scene.

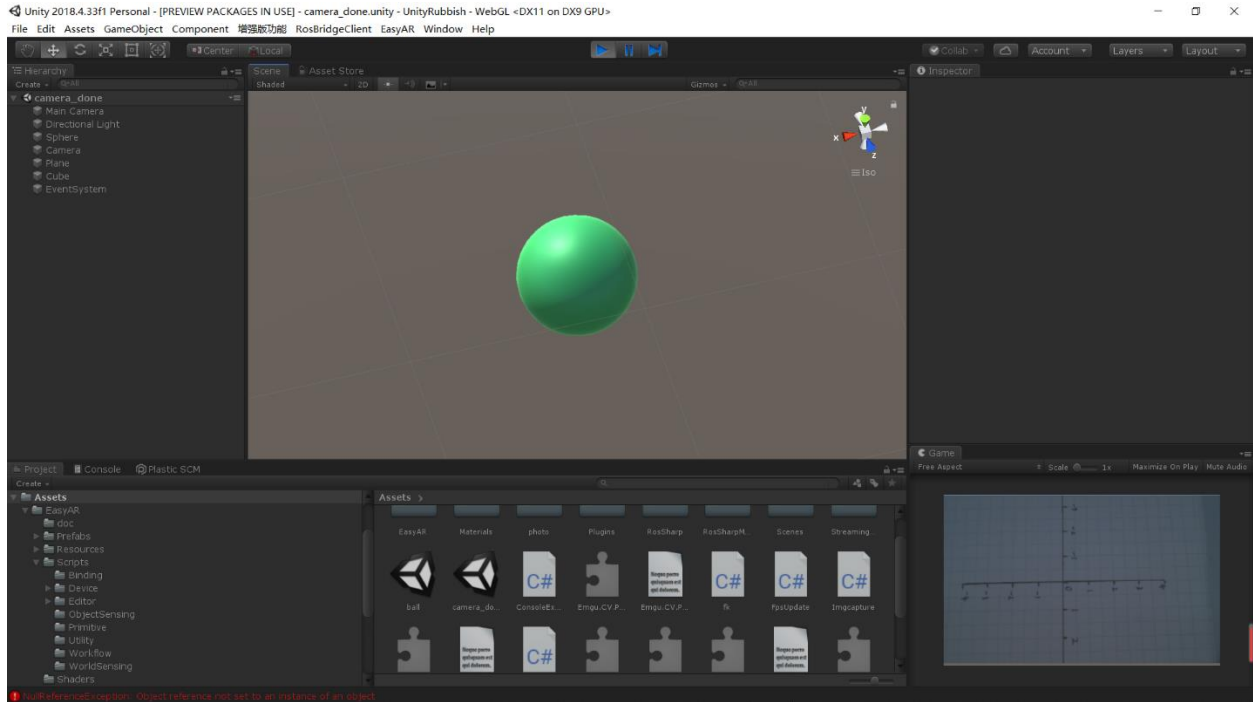


Figure 6.2. Unity3D scene

Figure 6.3 shows the Unity Game Window Outcome. As for the verification of “Imgcapture” C# script, after we push the “play” button in Unity 3D, we can see that the endoscope video information is shown in the Game Window indicating that we successfully receive the endoscope video information into the Unity 3D scene and there exists a red circle with a dot in its center point indicating that we successfully get the virtual camera video information and use it to accomplish the AR tagging function.

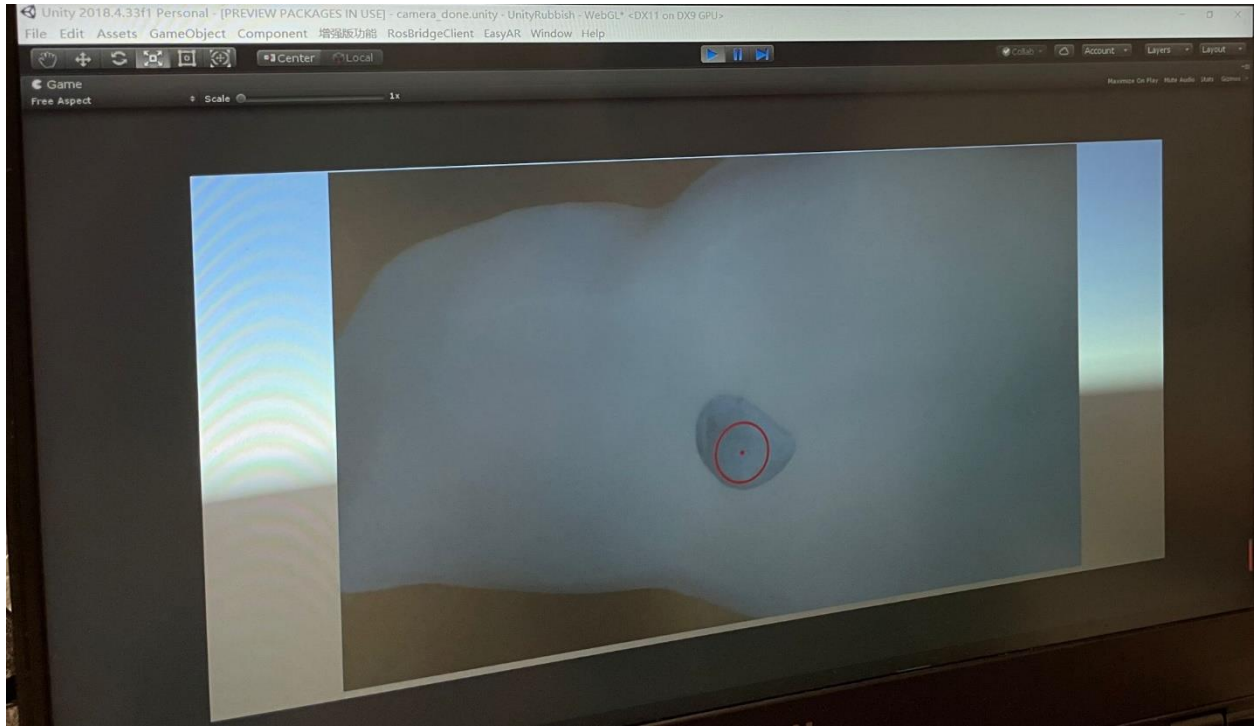


Figure 6.3. Unity Game Window Outcome

For the movement regulation of the virtual camera, its location and orientation are computed by the forward kinematics algorithm. By tracking the joint state information from the published and subscribed ROS messages, the virtual camera simulates the movement of the endoscope embedded on the robot arm, with latency less than 3 seconds. Figure 6.4 shows the panel indicating the location and the orientation of the virtual camera during the operation, which is updated automatically whenever we send ROS instructions in that change the pose of the robot arm.

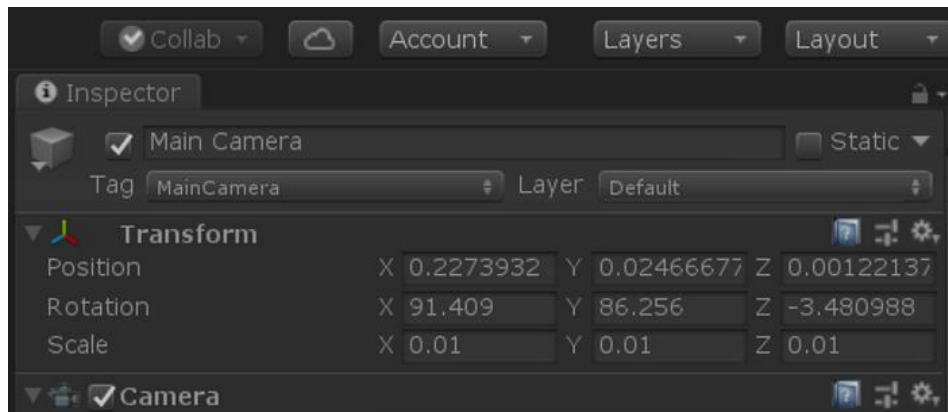


Figure 6.4. Control Panel of the Virtual Camera During the Operation of the Robot Arm.

4 Costs

The average starting salary per year for graduate ECE students of UIUC was \$84250 in 2014-2015 [6]. Assuming that the working hour per year is 1980 hours, the hourly wage would be \$42.6.

In that case, the labor cost for our project for each partner would be:

$$\$42.6/\text{hour} \times 100 \text{ hours} = \$4260 \quad (9)$$

Table 1 Costs for All Components

Component	Description	Manufacturer	Part #	Quantity	Cost (USA\$)
PC	Image processing and GUI	Dell	1	1	1,599.99
Robot Arm	Control endoscope position and orientation	ROBOTIS	2	1	1,199.00
Switch Adapter	Convert 220V AC to 12V DC	DVE	3	1	15.99
OpenCR Board	Control robot arm pose	ROBOTIS	4	1	179.90
Endoscope	Collect images	PENG-DU	5	1	220.00
Liver Model	Simulate the liver for demonstration	3D printer	6	1	10.00
Connector	Align the endoscope with the robot arm gripper	3D printer	7	2	5.00
Screw & nut	Tighten the connectors	Kuanfine	8	one box	5.00
Labor	ECE & ME engineers	NONE	9	4	4,260.00
Total					20,278.98

5 Conclusion

5.1 Accomplishments

ROS communication between Ubuntu Virtual Machine and Unity is accomplished using Ros Bridge Client, which is based on WebSocket protocol with latency the required value.

From the perspective of the projection, we successfully aligned the object, as well as the virtual main camera in the virtual scene to the relative location of the tumor and the endoscope in reality. As the endoscope is embedded on the robot arm, we keep track of the endoscope's location through the forward kinematics of the robot arm. And by setting the virtual camera to simulate the movement of the endoscope, a label accurately tagging the location of the tumor is successfully produced on the image returned by the endoscope.

In conclusion, we design a system that is able to produce an in-time visualization of the tumor, once the tumor is previously located by advanced medical imaging techniques. By visualizing the location on the image returned by the endoscope, the information from different sources is successfully integrated into the same physical space to assist the physicians.

5.2 Uncertainties

Upon successful projection of the location of the tumor in the image returned by the endoscope, there are several issues that remain to be solved.

Limited computational capacity of Unity3D C#: The projection algorithm involves massive matrix calculations that increase the burden of the Unity Control GUI. As its computational capacity is limited, the latency of the system proves to be not so well. Besides, the data transmission between the ROS control GUI and the Unity3D GUI also causes some accuracy and latency issues.

5.3 Ethical considerations

There exists plenty of safety hazards in our project, considering its nature of being an electronic medical system. The first one would be the danger that lies within cybersecurity. In our project, as the surgery is conducted by a robot arm, cyber-attacks on the controlling systems would cause severe consequences, which is a direct violation of IEEE Code of Ethics #1: "to hold paramount the safety, health, and welfare of the public" and #4: "To avoid unlawful conduct in professional activities." [7] To solve this, we propose that the medical system designed by our project should be embedded with strong cryptosystem, and should be operated only in an off-line mode, except for behaviors like system upgrade.

The second one would be the privacy offense issues. As our project involves imaging systems, there is a risk that the patient's digital data recorded by the imaging systems are used for other purposes against the patient's interest. According to the AMA Code of Medical Ethics #3.3.2: "Information gathered and recorded in association with the care of a patient is confidential, regardless of the form in which it is collected or stored." [8] To address the problem, we only keep the records in our database under the patient's permission. And we will restrict the data entry and access to authorized personnel by creating

password for our database. The access records will be audited routinely to prevent the remote attack by hacker.

The third one would be the potential operating error. The robot arm that conducts the surgery requires exceptionally high accuracy and stability. However, there are still risks of engaging mechanical faults such as misplacement. In that case we need certain security mechanisms to prevent the system from causing more damage to patients, which is an implementation of IEEE Code of Ethics #1: “to hold paramount the safety, health, and welfare of the public” [7] In our project, we propose an “emergency stop” mode to our system, which can be triggered by a single click of a button. Once the “emergency stop” mode is activated, the robot arm would halt and remain still immediately, waiting for further recalibrations to reactivate the system.

The fourth one would be the potential damage to the system when being operated in the liquid environment of abdomen. The ISO 8600-1:2015 states the general requirement of endoscope: “Depending on their utilization, endoscopes devices shall be provided with fittings/connectors for input or output of liquid or gaseous media.” [9] In our system, we choose waterproof fittings for the endoscope part that will be inserted to the abdomen to avoid potential damage caused by the liquid in abdomen. Moreover, for the connectors between the endoscope and computer, we will cover the connectors with waterproof material in case of possible splash of abdomen liquid.

The final potential risk is that bacteria may infect patients through components operating in the abdomen. The ISO 13485 for medical devices mentions: “Medical device intended to meet the requirements for sterility.” [10] To solve this risk, we set strict hygiene policies for our system. Before any components enter patients’ body, it must be sterilized and stored in aseptic conditions. And the interior of these components should also be an aseptic condition which prevent the breed of bacteria.

5.4 Future work

The auxiliary system is merely a primitive prototype and there still exists potential for improvement. The OpenManipulator-X robot arm we use only have 5 DOFs and due to that, the endoscope can only move within limited directions. In future improvement, we can replace the OpenManipulator-X robot arm with more advanced high DOF robot arm and redesign the forward kinematics algorithm for the new robot arm to make the auxiliary system adapt to the new robot arm. With the higher DOF robot arm, the endoscope will be given more freedom to move and provide convenience for the doctors. Moreover, currently we only draw a 2D tag on the endoscope image. It does not provide any detailed information about the tumor except for the location. To further improve the system, we can substitute the 2D tag with the projection of the tumor 3D model which will provide useful information, for example, vascularity, for the doctors and reduce the difficulty and risk of the surgery.

References

- [1] Reynisson, P. J., Leira, H. O., Hernes, T. N., Hofstad, E. F., Scali, M., Sorger, H., Amundsen, T., Lindseth, F., and Langø, T., Navigated bronchoscopy: a technical review. *Journal of Bronchology & Interventional Pulmonology* 21 (3): 242–264, 2014.
- [2] Burdall, O. C., Boddy, A. P., Fullick, J., Blazeby, J., Krysztopik, R., Streets, C., Hollowood, A., Barham, C. P., and Titcomb, D., A comparative study of survival after minimally invasive and open oesophagectomy. *Surg. Endosc.* 29 (2): 431–437, 2015.
- [3] Yu, F., Song, E., Liu, H. et al. An Augmented Reality Endoscope System for Ureter Position Detection. *J Med Syst* 42, 138 (2018). Available at: <https://doi.org/10.1007/s10916-018-0992-8>.
- [4] P. Vávra, J. Roman, P. Zonča, P. Ihnát, M. Němec, J. Kumar, N. Habib, A. El-Gendi, “Recent Development of Augmented Reality in Surgery: A Review”, *Journal of Healthcare Engineering*, vol. 2017, Article ID 4574172, 9 pages, 2017. Available at: <https://doi.org/10.1155/2017/4574172>.
- [5] ROBOTIS, “OpenMANIPULATOR-X e-Manual”, 2021. Available at: https://manual.robotis.com/docs/en/platform/openmanipulator_x/overview. Accessed March 2021.
- [6] ece.dev.engr.illinois.edu, “ECE GRADUATE STARTING SALARIES”, 2021. Available at: <https://ece.dev.engr.illinois.edu/admissions/why-ece/salary-averages.asp>. Accessed March 2021.
- [7] ieee.org, "IEEE IEEE Code of Ethics", 2021. [Online]. Available at: <http://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed: 4-March-2021].
- [8] ama-assn.org, “Code of Medical Ethics”. [Online]. Available at: <https://www.ama-assn.org/delivering-care/ethics/confidentiality-electronic-medical-records>.
- [9] iso.org, “ISO 8600-1:2015(en) Endoscopes - Medical endoscopes and endotherapy devices - Part 1: General requirements”. [Online]. Available at: <https://www.iso.org/obp/ui/#iso:std:iso:8600:-1:ed4:v1:en>.
- [10] iso.org, “ISO 13485:2016(en) Medical devices - Quality management systems - Requirements for regulatory purposes”. [Online]. Available at: <https://www.iso.org/obp/ui#iso:std:iso:13485:ed3:v1:en>.

Appendix A ROS Topic and Message Table for Robot Arm

Table 2 OpenManipulator’s Topics and Messages

Published Topic List	/open_manipulator/joint_states /open_manipulator/gripper/kinematics_pose /open_manipulator/states
Subscribed Topic List	/open_manipulator/option
Service Server List	/open_manipulator/goal_joint_space_path /open_manipulator/goal_task_space_path /open_manipulator/goal_task_space_path_position_only /open_manipulator/goal_task_space_path_orientation_only /open_manipulator/goal_joint_space_path_from_present /open_manipulator/goal_task_space_path_from_present /open_manipulator/goal_task_space_path_from_present_position_only /open_manipulator/goal_task_space_path_from_present_orientation_only /open_manipulator/goal_tool_control /open_manipulator/set_actuator_state /open_manipulator/goal_drawing_trajectory

Appendix B Requirement and Verification Table

Table 3 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Take 100 – 240 V AC (Free voltage) as input and provide 7 – 24 V output continuously.	1. a. The DVE Switching Adapter provides 12 V input to OpenCR board through SMPS In port. b. Use voltmeter to make sure the input voltage of DVE Switching Adapter is within 100 – 240 V. c. Use voltmeter to make sure the output voltage of DVE Switching Adapter is within 7 - 24 V.	Y
1 The OpenCR board should implement communication	1. Launch the OpenManipulator control GUI of ROS package. After press “Timer Start” button, The	Y

<p>between PC and robot arm with time latency less than 0.8 s.</p> <p>2. Output 10.0 - 14.8 V power sources to robot arm.</p>	<p>Logging screen will show the QTimer latency of 100ms.</p> <p>2. Use voltmeter to make sure the output voltage is 10.0 - 14.8 V.</p>	
<ol style="list-style-type: none"> 1. Check status of five actuators to ensure they are turned on normally before launching the robot arm. 2. Payload of at least 500 g to support the weight of endoscope and connector. 3. Input voltage between 10.0 - 14.8 V. 	<ol style="list-style-type: none"> 1. <ol style="list-style-type: none"> a. Implement “roslaunch” instruction to turn on 5 XM430-W350-T Actuators (ID 11, 12, 13, 14, 15). b. Give authority to access /dev/ttyACM0 port using “sudo chmod a+rm” instruction. c. Check the actuator status using “rostopic” instruction through port /dev/ttyACM0 of OpenCR board. 2. Put 500 g weight on the top of gripper to test the payload. 3. OpenCR board can provide 12 V voltage output through Dynamixel-TTL. Measure the input with a voltmeter to make sure the voltage is 10.0 - 14.8 V. 	Y
<ol style="list-style-type: none"> 1. The size of the gripper should be 44 × 17 × 15 [mm] to fit the size of the gripper. 	<ol style="list-style-type: none"> 1. After 3D printing the connector, use electronic digital calipers to make sure the size of 44 × 17 × 15 [mm]. 	Y
<ol style="list-style-type: none"> 1. The camera has a resolution of 1920 x 1080 pixels and a frame rate of 30 fps. It has manual focus facility and can support Windows operating system 2. The position of the endoscope can be controlled by the robot arm with an error tolerance of 0.005 m in the 3D map. 	<ol style="list-style-type: none"> 1. <ol style="list-style-type: none"> a. Connect the endoscope to a PC. The SmartCamera software has the option to set frame rate to 30 fps. b. Write codes of the function to test the frame rate and show it on our Unity AR Processing GUI. 1. <ol style="list-style-type: none"> a. The control interface would calculate and return the distance between the 3D position we set and the actual position of the endoscope once per 100 ms (indicated by OpenCR GUI tool). b. We are supposed to set complementary offsets to our control algorithm to make sure the average distance per second is less than 0.005 m. 	Y