Project 3 Proposal

Robot System Control for Automating Mosquito Microdissection

EN 601.656 Computer Integrated Surgery II

Zhuohong He (zhe17@jhu.edu)

February 25, 2021

Contents

1 Introduction

Malaria vaccine production and distribution is a critical public health challenge for many developing countries. Sanaria Inc is working with the Johns Hopkins University Laboratory for Computational Sensing & Robotics (JHU LCSR) to develop a robotic system to automate mosquito saliva gland collection, a bottleneck in vaccine production. Previous researchers at JHU have successfully extracted salivary glands by developing the proper robot hardware, control algorithm, and computer vision models. My goal will be to implement improvements to the existing control algorithm to increase parallelization, add error recovery, and support the next generation hardware and computer vision efforts.

2 Background and Significance

The malaria parasite is response for a global disease affecting 228 million individuals and causing 405,000 deaths in 2018 worldwide. Malaria is most prevalent in underdeveloped countries where controlling the spread of the virus is a national public health challenge. The parasite is transmitted through mosquitoes which ingest blood from multiple people. Prior to transmission, the sporozoite version of the parasite resides in the mosquito's salivary gland. To curb infection, Sanaria Inc is developing a vaccine which has proven to be highly effective with long-lasting protective effects against malaria. However, extracting the salivary glands from recent deceased mosquitoes is one remaining bottleneck currently performed by human operators. If extraction is automated, Sanaria will be able to significantly reduce the infection rate across the globe and save the lives of many individuals living in areas where malaria is prevalent.

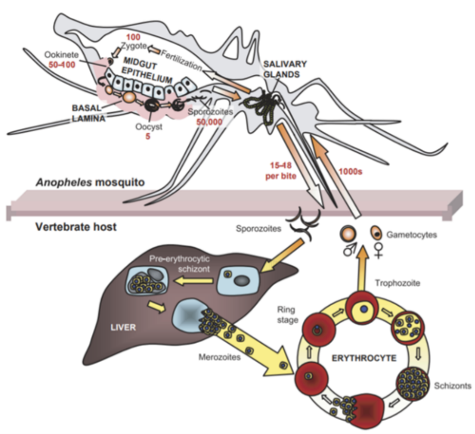

Figure 1: *Malaria life cycle. Note the position of the salivary glands within the mosquito upper abdomen and its function of storing the sporozoite immediately before infection into the vetebrate host.[1]*

3 Prior Work

Two systems have been developed to automate mosquito dissection. First, M. Schrum et al. developed the semi-autonomous mosquito microdissection system (sAMMS) (fig 2) in 2018 which still required a human operator albeit increasing the dissection rate from 300 mosquitoes per hour (mph), to 430mph. Our goal dissection rate is 600mph. [1]

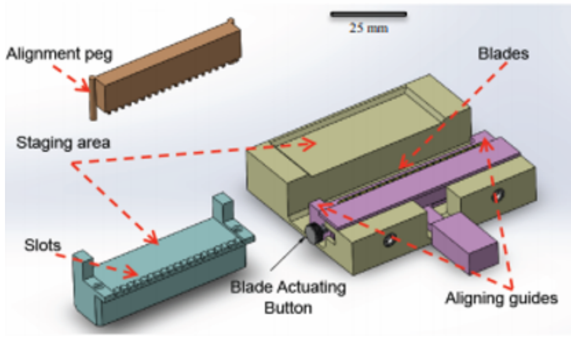

Figure 2: *The semi-autonomous mosquito microdissection system (sAMMS) which was tested for use by trained human operators at Sanaria Inc. [1]*

A second system started by H. Phalen et al. in early 2020 focuses on full automation through a 4 degree of freedom (DOF) robot manipulator with a controlled gripper as the end-effector. [2] In another paper under review for ICRA 21, the team at JHU has incorporated further upgrades to Phalen's work. New hardware including a squeezer, turntable, and cleaning station were added to increase the completeness, autonomy, and speed of the system. This system starts with the 30-40 mosquitoes in a center dish (fig 3. Next, an operator drags mosquitoes onto the turntable's loading area. The turntable then rotates the mosquito under the overhead camera. The robot and gripper uses the camera-fed convolution neural network (CNN) output to execute a placement between the blades of the cutting apparatus. The apparatus then decapitates the mosquito and the cartridge moves the mosquito to the squeezing station which applies pressure to the abdomen and extrudes the salivary gland from the neck. Our experiments demonstrate that one mosquito is processed every minute. This speed is significantly lower than our 600mph goal.

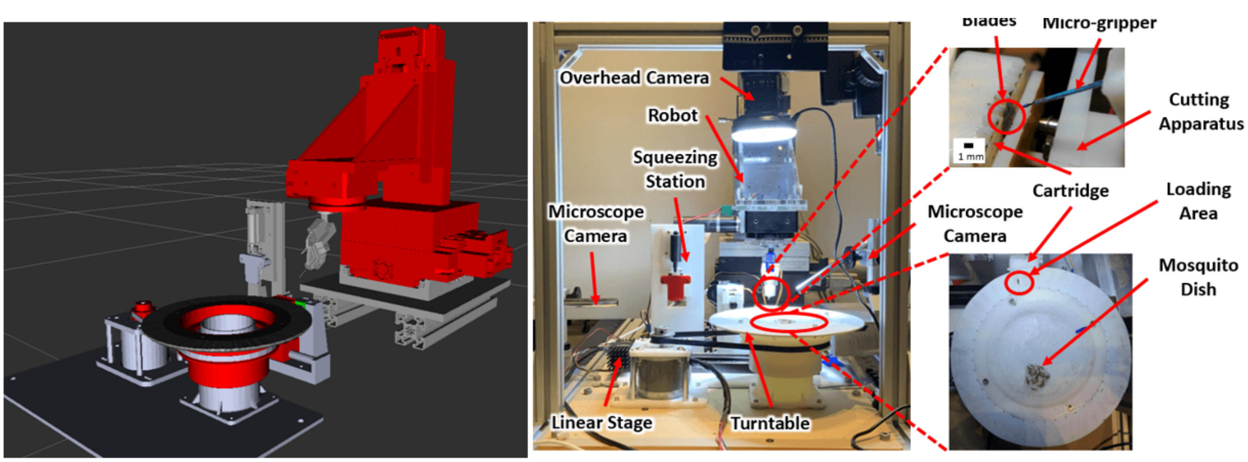

Figure 3: *The simulator (left) and the physical system (right) for the paper under review for ICRA21*

Our team has also built a simulator (fig 3) for the physical system which interfaces with the control algorithm. Using the simulator, we are able to specify the error rate and safely & remotely test the robot control algorithm.

4 Goals

Significant progress has been made after the submission of the ICRA21 paper. Our new efforts focus on designing hardware with independent subsystems that can execute some tasks in parallel. Thus, the goal for my Computer Integrated Surgery II project is to develop a next generation control algorithm that includes parallel process, error checking, and error recovery. Furthermore, I will help support the development of new hardware and CNN models, and I will conduct control system testing using the simulator and the hardware. Specifically, my deliverables are documented in figure 4.

	a. Minimum Deliverable	**b. Expected Deliverable**	**c. Maximum Deliverable**
1. The Basics	Fully debugged, well documented controller; Create five independent nodes to call services to execute flowchart steps.	Create advanced figures and documentation in the README files and wiki.	Future-proof the system by writing highly generalizable and inheritable code. Create abstract objects, etc.
2. Error & Recovery	Implement the errors & recovery shown in the flowchart. Define / document other errors.		Work with CV team to implement further error handling; one example is dragging detection. Another is ensuring neck is between blades.
3. Integration	Controller operates on the new hardware / setup. Focusing on using placeholders instead of Low-Level Arduino/Galil control.	Write Arduino/Galil code for low level control of hardware. Create working services for the physical robot.	
4. Testing	Visual testing using the simulation or hardware. Component Testing.	Testing on select hardware on the physical system.	Quantitative testing on the physical system to determine success rate.

Figure 4: *The deliverables for this project by category and deliverable level.*

5 Technical Approach

We are using an open-source framework called Robot Operating System (ROS) to implement our robot controller. ROS is a collection of tools and libraries that simplify the complex task of robot control. At the core of ROS is the node, a logical unit that can communicate with other nodes using ROS messages and execute a script to move the robot, call a neural network, output to a GUI, and more.

In order to structure the code to allow for error recovery and ease of communication, we are employing a ROS package known as actionlib. Actionlib's *ActionClient* and *ActionServer* communicate via a "ROS Action Protocol" which is built on top of basic ROS messages. The client is the caller which requests the server, or the callee, to execute a task and return the result. Thus, the client-server architecture provides the basic functions to efficiently operate the robot system (fig 5).

Our lab has already implemented a similar controller without error recovery for our last hardware iteration using the client-server architecture. My project will focus on supporting

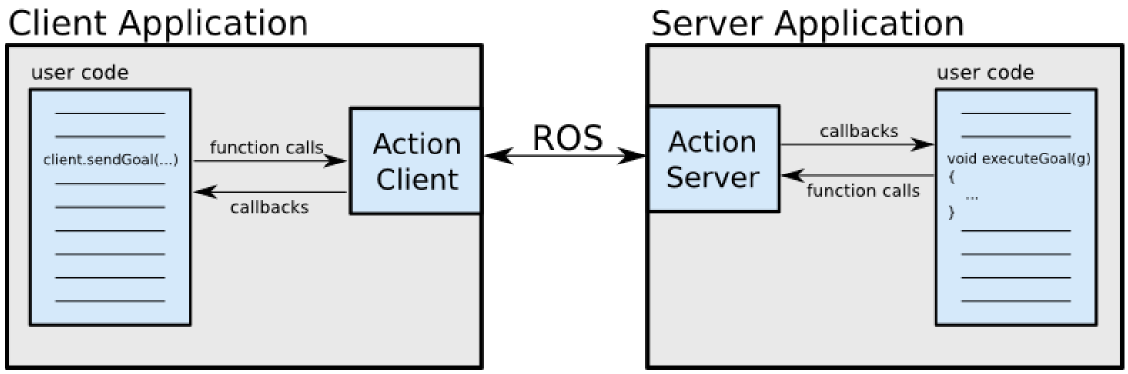

Figure 5: *Two application communicating through an ActionServer and ActionClient pair.*

the new hardware, adding error recovery, and parallelizing processes. Our new controller will be based on a flow chart (fig 6) which has five parallel nodes which each control independent subsystems: the Mosquito Pick-Place-Decapitate robot (MPPD), gland extractor, turntable cleaner, turntable, and robot cleaner.

Our lab has two means for debugging and testing. The first option is testing on the physical robot which is located in JHU Laboratory for Computational Sensing and Robotics (LCSR) robotorium and can be accessed and operated remotely as well. The second option is an ROS RVIZ simulation which I helped develop for the last hardware iteration. It needs to be updated to the newest hardware in order to be used for our purposes.

6 Testing Plan

Our testing plan is composed of three levels: component testing, system testing through the simulator, and system testing on the hardware. Each level introduces more complexity and passing all three will result in a successful deliverable.

6.1 Level 1: Component Testing

Component testing will occur during the development of each node. I will write a script to command the node to execute both simple and edge actions to attempt to break the node. I will make improvements to the node if any issues arise. Ideally, component testing will occur on the simulator before the hardware. However, we may need to test directly on the hardware if development of the simulator is delayed (see fig 8).

6.2 Level 2: System Testing on Simulator

System testing on the simulator consists of creating scenarios and generating a combination of errors on the simulator. The scenarios can be created by sending static images to the CNN model to simulate the overhead camera's image captures. Our simulator currently has the ability to generate hardware and situational errors at a specified probability rate. Thus, we can test the controller's ability to recover from unlikely errors and rare error combinations.

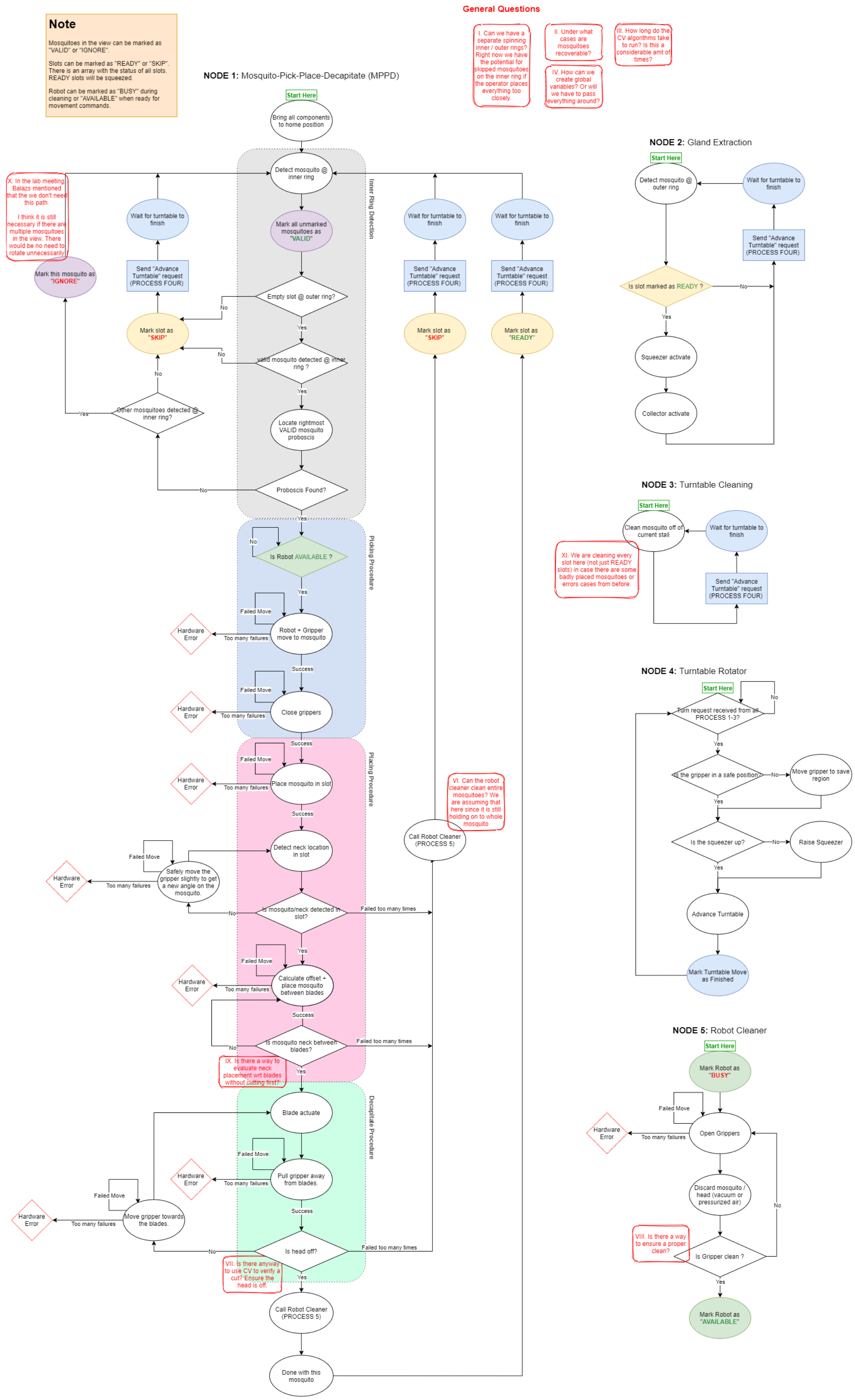

Figure 6: *Full flowchart detailing 5 nodes: Mosquito Pick-Place-Decapitate (MPPD), gland extractor, turntable cleaner, turntable, and robot cleaner. A magnify-able flowchart can be found at:* [Sanaria Controller Flowchart]

6.3 Level 3: System Testing on Hardware

System testing on hardware will accomplish the same error recovery testing as level 2 and help determine the mosquito throughput rate measured in mosquitoes per hour (mph). We can compare the mph to previous controllers and systems to determine whether the parallel controller with error recovery contributes to a rate gain. Level 3 testing will depend on developments from the hardware and computer vision teams (see fig 8) because an incomplete system will not have a throughput rate.

7 Timeline and Dependencies

The proposed timeline for the project (fig 7) shows the weekly goals and the deadline for deliverables specified in fig 4. If we work as planned, we will reach our minimum deliverable by April 12th, 2021 and achieve our expected deliverable by April 26, 2021.

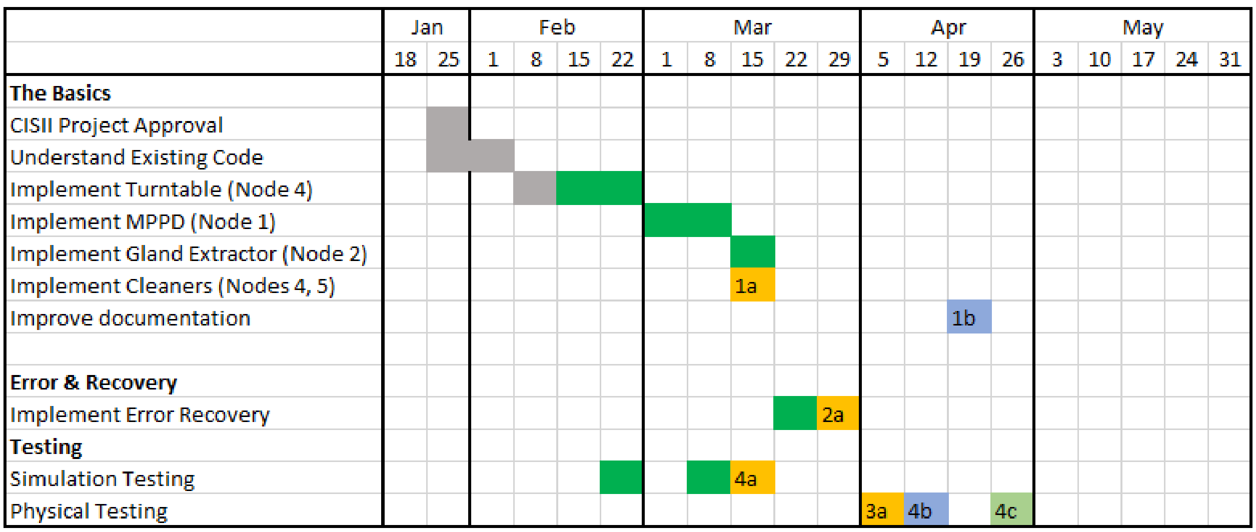

Figure 7: *Development timeline with deliverable deadlines*

Because I am working closely with the hardware and computer vision teams in the lab, there are also several dependencies we must consider. Figure 8 summarizes the dependencies, the need justification, the current status, follow up next step, contingency plan, and resolution deadline.

8 Management Plan

8.1 Team Members and Roles

This team consists of:

- **Zhuohong (Zooey) He** (zhe17@jhu.edu)
  MSE Robotics, Laboratory for Computational Sensing & Robotics, second year
  Sole responsibility for all tasks required for this project.

The mentors consist of:

Dependency	Need	Status	Followup	Contingency Plan	Deadline
Swipe Access to Robot in Lab	Testing and Debugging during development	Resolved	N/A	Update and test through simulation instead	2/1/2021
Error CNN from Computer Vision team	Used for initiating some recovery paths in flowchart	In Progress	Contact and provide details for CV team	Use placeholder services for error detection	3/22/2021
Server/Client Structure is Insufficient	We use this structure to ensure robust communication	N/A	Understand more about ActionLib	Develop a node without ActionLib, or override functions to support needs.	3/22/2021
JHU Remains in semi-open state	Needed to conduct testing on the physical hardware	N/A	Be vigilant to changes, and keep an eye out for JHU COVID status	Update and test through simulation instead	4/20/2021
Simulator needs to be developed	The simulator is an important bridge to hardware testing	In Progress	Follow-up with another lab member (Wanze Li) who is working on this	Create a cautious plan to move to the hardware immediately	3/15/2021

Figure 8: *Dependencies, the need statement, current status, and contingency plan.*

- **Dr. Simon Leonard** (sleonard@jhu.edu)
  Assistant Research Professor (Computer Science)
  Dr. Leonard is very experienced in ROS development and will help advise on the software architecture design, error recovery logic, and resolving difficult ROS challenges. He will be the primary mentor on this project.

- **Dr. Russell Taylor** (rht@jhu.edu)
  John C. Malone Professor (Computer Science)
  Dr. Taylor has a decorated background in medical robotics and will provide his expertise and leadership to ensure our development is aligned with the team and larger research goal. Dr. Taylor will be a valueable resource and a second mentor on this project.

Our contacts at our industry partner Sanaria include:

- **Dr. Kim Lee Sim**
  Executive Vice President Process Development and Manufacturing, Sanaria Inc.

- **Sumana Chakravarty**
  Sanaria Inc.

8.2 Meeting Schedule and Communication Platforms

Scheduled meetings for this project will occur on a weekly basis and serves to update the mentor and the larger research group.

- **Friday 11:30am - 12:00pm:** Update meeting between Zhuohong He and Dr. Simon Leonard. The purpose of this meeting is discuss technical challenges faced during the week and coordinate efforts to move according to the timeline.

- **Tuesday and Thursday Evening:** Reserved time to run tests on the physical robot. This time can also be used for extra meetings with Dr. Leonard or Dr. Taylor.

- **Monday 10:00am - 11:00am:** Update meeting with the research group. The purpose of this meeting is to coordinate efforts with the computer vision and hardware teams, to hold each other accountable, and address concerns.

Communication between Zhuohong He and Dr. Simon Leonard (primary mentor) will occur largely through email. However, efforts will be made to exchange phone numbers for faster communication.

9 Reading List

References

[1] Mariah Schrum, Amanda Canezin, Sumana Chakravarty, Michelle Laskowski, Suat Comert, Yunuscan Sevimli, Gregory S. Chirikjian, Stephen L. Hoffman, and Russell H. Taylor. An efficient production process for extracting salivary glands from mosquitoes, 2019.

[2] Henry Phalen, Prasad Vagdargi, Mariah L. Schrum, Sumana Chakravarty, Amanda Canezin, Michael Pozin, Suat Coemert, Iulian Iordachita, Stephen L. Hoffman, Gregory S. Chirikjian, and Russell H. Taylor. A mosquito pick-and-place system for pfspz-based malaria vaccine production, 2020.