Group 13: An Improved GUI and Visual Navigation of the Robo-ELF
600.446 Computer Integrated Surgery II, Spring 2012
Final Paper

Team Members: Jonathan Kriss
Mentors: Kevin Olds, Dr. Russ Taylor
Collaborators: Renata Smith, Dr. Jeremy Richmon

I. Background

There are approximately 25,000 annual cases of throat cancer in the United States, about 6,000 of which are fatal. Radiation and chemotherapy are common treatment methods, but surgery is often required to remove late-stage tumors from the throat. There are two options in performing this surgery, intra-airway or open throat. Intra-airway is the preferred method when possible because it is much less invasive and results in much faster recovery times. When performing intra-airway surgery, an endoscope is used to provide visualization of the larynx. Rigid endoscopes are useful because they can be positioned and held in place with a stand. However, they offer limited range of vision and cannot see at all certain areas of the sub-glottal region. Therein lies the advantage of flexible endoscopes. They provide a much larger range of vision than rigid endoscopes. The problem is that there are no existing mechanisms to hold a flexible endoscope once it has been positioned, so another surgeon must be present to hold and control the endoscope with two hands while the other performs the operation.

The Robotic EndoLaryngeal Flexible Scope(Robo-ELF) solves this problem. It provides a means to hold and position a flexible endoscope during surgery so the surgeon has both hands free to operate. A prototype robot was built and tested in 2011 using phantoms and human cadavers [1]. It functioned well in testing and surgeons were pleased with the results. The ultimate goal of the project is to enter human clinical trials with the Robo-ELF. Completing the requirements for FDA approval of clinical trials was the main goal of this project.

Requirements for FDA approval of clinical trials include a formal risk assessment and mitigation process, documentation of safety features and their validation, and a detailed user manual for the system including setup, breakdown and operational instructions and explanations of error codes. Completing these requirements necessitated several design changes to the structure of the robot and to the software system. The robot needed to be made easier to setup and take down, and the software safety error handling needed to be more robust. The robot required a full draping and cleaning procedure, and a new, more user-friendly Graphical User Interface(GUI).

To determine exactly what changes needed to be made to complete these requirements and how to best implement them, we held several system review meetings with LSCR faculty, our surgeon and the students involved in the project. There were also software-specific meetings to conduct a code walkthrough and review all of the software systems. The outcome of these reviews was recommendations for the changes made throughout the semester. The most major software recommendations were to improve error identification and handling and to improve the GUI.

II. Technical Approach

The most important factor in securing FDA approval for the Robo-ELF is ensuring its safety. There are a number of safety features built into its design and many safety features implemented through hardware and/or software in the system. The physical design of the robot limits its motion to a non-threatening range and the software commands it to move only at safe speeds. The only part of the system that touches the patient is the endoscope shaft, which is already approved for surgical use. The Robo-ELF can do no more damage to a patient than a surgeon operating the same endoscope. Even with these design safety measures, we implemented other safety mechanisms to ensure that the surgeon is in full control of the robot at all times and that the robot will deactivate if there is a chance that it will go out of control.

Many of the safety mechanisms rely on proper use and maintenance of the system. This means that the user manual must include detailed use and care instructions. We included step-by-step instructions for setup and takedown, important safety checks, and safe operating instructions. The most significant risk is for the robot to fall on top of the patient. The robot is attached to the bedrail of the surgical bed during setup, and it could fall and injure the patient is if it is set up improperly. The detailed user manual seeks to eliminate this risk. Regular safety checks to ensure its structural stability are also recommended. The second most significant risk is for cross-patient contamination through the Robo-ELF. As previously stated, the only part of the system that touches the patient is the endoscope, for which there is already an established cleaning procedure which will be followed. However, droplets from splash or breathing could contaminate the robot. To prevent transmission of pathogens from one patient to another and to reduce the risk of infection, we developed a draping procedure for the system, which is also detailed in the user manual.

The software system for the Robo-ELF is built using the CISST libraries, written in C++. It contains four major component classes, `devGalilController`, `robotTask`, `qtDisplayTask`, and an `svlVideo` stream object. `devGalilController` is the CISST C++ wrapper for the Galil motor controller. It contains functions to send commands to, and receive feedback from, the motor controller. It is the low-level software connection between the PC and the motor controller. `robotTask` is the high-level software interface between the PC and the robot. It contains an instance of the `devGalilController` class along with all of the functions to implement safety and control algorithms for the system. `qtDisplayTask` is the GUI portion of the system. It contains a Qt class, `throatGUI`, that implements the display structure. It also contains CISST multitask functions to receive data from `robotTask` and display it in realtime on the GUI. The `svlVideo` stream displays the scope view on the screen and records the video for future playback.

The hardware of the system consists mostly of the robot itself and its associated control boxes. In addition to the robotic arm that holds the endoscope, there is a box to house the motor controller, a box to house joysticks and an emergency stop switch, the endoscope control box, and a PC running Windows7. Integrated into the robot arm are three motors with digital encoders and analog potentiometers mounted to each of three active degrees of freedom. Each axis of motion also includes forward and reverse limit switches that, when tripped, will stop motion on that axis. The robot cannot physically move past the limit switches, so their failure does not present a danger to the patient. Their main purpose is to provide homing limits and prevent damage to the robot. The emergency stop switch is wired to the motor power. Flipping the switch will disable all of the motors. It can be activated by pressing a physical button or through a

software-activated relay. The PC is connected to the motor controller via Ethernet cable and to the emergency stop by USB.

III. Results and Discussion

We completed several software reviews to determine what improvements needed to be made to prepare the system for clinical use. The most important change was a need for centralized, systematic error handling. We determined that errors should be categorized into serious and non-serious groups. Serious errors are errors that pose a significant danger to the surgeon or the patient. They require the system to be stopped immediately and removed from any procedure until maintenance is completed. Non-serious errors do not pose an immediate threat to patient or surgeon. They require a system restart and a recalibration to make the system safe.

With the help of LCSR faculty members and graduate students, we completed a Failure Mode Effects Assessment(FMEA) for the Robo-ELF. This document was requested specifically by the FDA and ensures that we have considered every possible system failure and its potential danger to the patient. We determined that our safety mechanisms adequately mitigated the possible dangers. The Robo-ELF has enough safety mechanisms and redundant sensors to prevent any injury to patients or surgeons.

Most of the safety mechanisms are present in the software. They include: a software-activated emergency stop switch, a watchdog timer between the PC and the motor controller, redundant checking between motor encoders and potentiometers, and hardware and software motion limits. The emergency stop immediately cuts power to the motors and can be triggered by the software system or by pressing a large red button. The watchdog timer runs on both the motor controller and the PC, checking that each one can communicate with the other. There are digital encoders on the motors and analog potentiometers attached to each axis of motion. The values read from each of them are continuously compared to verify that the encoders are functioning properly. If any of the software safety systems show an error, or the Galil controller shows a failure, the emergency stop is activated and an error message displayed. This message contains information about the error and the proper course of action to take to recover from it. All of the software changes and updates are documented in the code itself and in high-level documents and diagrams. The code is available in an online repository (https://svn.lcsr.jhu.edu/robotorium/trunk/apps/ThroatRobot/)

There were significant changes made to the error handling and detection in all pieces of the software. Previous to the recent changes, errors were not adequately passed from the Galil to the PC program, and PC-based safety checks were implemented unsystematically without logs or detailed error messages. We reorganized and restructured the `robotTask` and `devGalilController` classes to the following specifications. All safety failure detections are reported through exceptions, implemented using CISST's `cmnThrow` function which also logs the errors to a file. All exceptions inherit from the exception class `RobotException` which is defined in `devGalilController`. Within the `devGalilController` class, several other classes of exceptions are defined to differentiate between types of errors.

Major error categories include: communication errors, motion errors, power errors. Communication errors relate to problems communicating between the PC and the motor controller, typically indicating a problem with the Ethernet cable. Motion errors indicate that there is problem commanding the robot to move. Usually this means there is a physical problem with the motors, encoders or joints of the robot. Power errors relate to the power supply for the

motors and controller. We consider communication errors to be non-serious because they can usually be fixed by reattaching or replacing a cable. Motion and power errors are considered serious because they could indicate a major problem with the robot that requires maintenance before safe reuse of the system. The error categories are differentiated by the class of exception that is generated. When exceptions are caught and handled, they are sorted by their sub-class, and appropriate action is taken for each category.

The safety features implemented in `robotTask`, the emergency stop, the watchdog timer, and encoder/potentiometer checking, are also reported through exceptions. Each check runs in its own function that is called once every periodic cycle of the task. If any of the checks fail, they throw an exception using `cmnThrow`, which logs the error and throws the exception. Exceptions are caught and rethrown by reference until they are caught in the main `Run` function of the task. There they are caught and the appropriate error message is printed. Exceptions generated by the `devGalilController` object in `robotTask` are also caught and handled in the `Run` function. If an error occurs during startup, it is caught and handled in the same manner by the `Startup` or `Configure` function. When any of the exceptions are caught, the emergency stop is triggered and a flag set to stop commanding motion to the robot. Error messages are displayed on the GUI and a more detailed error log is saved to a file. The program does not immediately exit, but will not issue new commands to the robot until it is restarted. This allows it to be safely shut down and a proper failure state to be recorded.

The emergency stop is connected to the PC via USB cable and communicates to it through a virtual serial port. The status of the connection is checked in each cycle of `robotTask` to verify that it is still connected. The relay circuit sends a response every time a command is sent to it, so while the robot is running, commands to activate the circuit are repeatedly sent. If one of these does not receive a response, an exception is generated and the PC will no longer send motion commands to the robot. The relay is powered through the USB connection, so if the cable is pulled out, the switch will close and cut power to the motors. The connection is checked continuously from startup to shutdown, and if a failure is detected, the robot will stop. A failure in the emergency stop communication is regarded as a non-serious error because it can usually be fixed by reattaching the USB cable. The software connection checks are to guard against failures other than a disconnected cable and will prevent motion if a failure is detected.

The watchdog timer consists of messages sent between the Galil and PC every 50ms or less. Each one sends a message to the other every 50ms and listens for messages from the other. If the delay between received messages is longer than 50ms, an error is detected and motion is halted. If the PC detects the disconnect, an exception is thrown and handled in the manner described above. If the Galil detects it, it enters an error state in which the motors are disabled. A watchdog timeout is considered a non-serious error because it can often be corrected by reattaching the Ethernet cable. If the cable is not the problem, the system will not be able to restart properly and will need to be sent for maintenance.

The implementation of the encoder/potentiometer checking was tested and improved this semester. At startup the system must rebuild tables of values from both the encoders and potentiometers to do cross-checking while the system runs. To build the tables, it runs the robot through its full range of motion in each axis, taking sample readings as it goes. The encoder and potentiometer values are matched by their array index. To do checking, it finds the value in the encoder table that is closest to the current value. Then it finds the potentiometer value at the same table index and compares it to the current value. If the potentiometer values are off by more

than an allowable difference an error will be generated and the robot will stop. That distance is calculated as: the maximum resolution, given the number of values, plus allowance for some noise and drift. To ensure that the system is working properly on startup and the table is not populated with bad values, newly generated tables are checked against a "gold standard" table, a set of values known to be accurate for the system. If the new values are not close to the standard table values, an exception is generated to halt startup and the system must be restarted or sent for maintenance.

We designed a new Graphical User Interface(GUI) for the system that is more useful in the OR than the current interface which was designed as a debugging tool in the lab. It includes a visual representation of the three motors and their position. The two rotational axes, the scope tip actuation and rotation about the scope axis, are represented by dials. The lateral motion along the scope axis is represented by a slider. This allows the user to know at a glance where they are in the robot's range of motion. This is a much easier system to read and understand than trying to glean the same information from encoder counts, previously the only position reference available to the user. The GUI displays fields to indicate the status of the limit switches. For each axis, an "S" or an "H" will be displayed when the software or hardware limit is tripped, respectively. System instructions and error messages are printed to the standard output window. We also reorganized the layout of the interface to eliminate the need for the user to drag multiple windows around the screen or switch between them to see all information.

IV. Management Summary

Jonathan Kriss completed all of the software changes and documentation with input from mentors and collaborators. Renata Smith and Kevin Olds completed structural changes to the robot required for clinical use, and Renata completed the non-software parts of the user manual. Meetings were held weekly among team members and two major meetings were held during the semester with mentors and faculty to assess progress and discuss design changes. Major lessons learned include: work will often require much more time than originally planned and it is very important to make sure work is done properly the first time. We ran into several electronics problems with the system that were the result of poor worksmanship in the original construction. These problems significantly delayed work on new goals. We also underestimated the amount of time it would take to complete our goals, and were not able to finish everything we had planned.

V. Future Work

While significant progress was made, we are not ready to submit our system to the FDA for review just yet. We plan to continue working over the next several weeks and should be ready within 5 to 6 weeks. Once the resubmission is made and permission granted from the FDA to continue, we will submit our IRB proposal and hopefully will be able to begin clinical trials by the end of the year. During this time, we will continue to upgrade and improve the system. We would like to implement vision-based navigation as we had planned to do this semester. There is also potential to include many other improvements involving more software functionality, the ability to integrate more sensors like ultrasound or OCT probes, and extending possible use cases to include longer endoscopes. A more intuitive, user-friendly control device is also desired. The current two-joystick system is usable, but a custom-designed single joystick would be preferred.

VI. Acknowledgements

Thanks to all collaborators and mentors for their hard work and assistance on the project. Special thanks to Min Yang Jung, Dr. Peter Kazanzides and Dr. Russ Taylor for their input on software and system design.

VII. References

[1] Olds, K., Hillel, A. T., Cha, E., Curry, M., Akst, L. M., Taylor, R. H. and Richmon, J. D. (2011), "Robotic endolaryngeal flexible (Robo-ELF) scope: A preclinical feasibility study." The Laryngoscope, 121: 2371–2374.