600.446: Advanced Computer Integrated Surgery

Final Project Report

Integration of LARS and Snake Robots

H. Tutkun Şen

Mentors: Prof. Russell Taylor, Paul Thienphrapa

May 19, 2011

**Abstract**

 In this report details of my class project for Advance Computer Integrated Surgery are described for which my goal was the software integration of JHU-Snake robot and JHU-LARS robot. For this purpose, the combined system is teleoperated using SpaceNavigator 3D mouse. In the design phase of this project, first an integration scheme is developed such that the combined system would be run with two different computers: LARS computer and Snake computer. In this approach LARS PC has all the logic and Snake PC acts as a slave robot responsible for implementing the FireWire controller only.

 The second phase of the project is the derivation of the combined kinematic equations of the system. In achieving this, some modifications and transformations are done on the existing kinematic equation of the system [1, 2]. After deriving the forward kinematic equations and the Jacobians of the combined system, the next job is the development of the advanced mode of control algorithms. For this approach, “Constraint Control Optimization Algorithm [3]” is used. Following to the implemented algorithm remote center of motion (RCM) and virtual wall operations are simulated and tested.

**Significance and Motivation**

 The motivation behind developing control software for the combined LARSnake system and implementing a teleoperative interface is to provide a versatile robotic research platform suitable for use in various medical applications. The redundant degrees of freedom and high mobility of the LARSnake robot make it well-suited for general clinical use. Currently, the integrated system is planned for real-time 3D ultrasound-based online registration of a dexterous surgical manipulator with verification using fluoroscopy in minimally invasive surgery approach.

In case of explosions and various similar incidents, some particles such as shrapnel or bullet fragments can get stuck in the heart and impede cardiac function. The conventional approach is removal of the foreign body through open heart surgery, which comes with high perioperative risk and long recovery time.

To solve this problem, a minimally invasive surgical system is proposed for removing the foreign objects from a beating heart. Also it is claimed that, using this approach can reduce the mortality risk, improving postoperative recovery, and potentially reduce operating room times. With the aim of solving this problem, the first thing to do is the integration of LARS Robot and Snake Robots in physical and software environments.

**Management Summary**

 At the beginning of the term, the aim was integrating both systems in a single Linux RT environment. However, I had concerns about the existing copies of the codes since they were both built using different versions of the CISST library. In the first week LARS code is updated to the current version of the CISST library, but it seemed very difficult to update the Snake code so another integration scheme is chosen and this scheme is followed in this project details of which will be given in Technical Summary part. There was no other change of plan took place during the design of the project such that, both robots are kinematically combined, constrained optimization algorithm is implemented and RCM mode of operation and simulation of virtual wall is implemented.

Figure 1

 For future work, for sure, the very first thing should be updating the Snake code to the current version of the CISST library and moving all the code to the Linux RTAI environment. Following this, calibration of the end-effector should be implemented such that for this purpose, Phantom Omni can be used instead of 3D Space Mouse. When all these are done, the system will be ready for medical experiments for the 3D Ultrasound Project.

 In this project primarily, familiarity with the CISST development framework is learned. Since all the robots in the JHU CIIS lab are programmed using this framework, it was very essential for me to gain some insight about the details of this structure. Secondly, an insight concerning kinematics and robot control methods is gained, moreover with the gained familiarity to the Constrained Optimization Algorithm is very crucial in understand moving principles of robots having redundant degrees of freedom.

**Technical Summary**

**System Overview**

 An overview of the LARSnake Teleoperation system is shown in Figure-2. The system can be thought of as divided into five main logical components:



Figure 2 LARSnake System Overview

1. The LARS Robot and its associated Galil motor controller/amplifier that provides low-level motion control,
2. LARS Slave Application that determines high-level motion commands,
3. The Snake Robot which is connected to the LARS Application through UDP
4. Snake Application responsible for low-level FireWire controller commands
5. The Master Controller Application that sends input from the user’s master device to the LARSnake application.

The LARS robot is a 7 degree-of-freedom (DOF) robot having 3 translation axes at the base and 3 rotations plus 1 translation axis at the tip. Snake consists of two serially connected snakes each having 2 DOF both of which are retains in two different planes.

 Since a position in 3D space is defined by 6 DOF, having 7 DOF LARS plus 4 DOF Snake makes the LARS robot spatially redundant. The motion control algorithm running in the LARS Application is designed to handle this redundancy.

**Software Overview**

To maximize portability and maintenance between the LARS and LARSnake code base, the organization of classes, tasks, and task interfaces for the LARSnake Application is little changed from the original LARS code. A simplified overview of its structure is provided in Figure 3, showing the major task and class names and the task interface. In the below Figure 3, red boxes shows the tasks and the task interfaces are drawn as arrows between the tasks. A “required” interface is represented by the tail of an arrow, whereas the head of an arrow represents a “provided” interface. Provided interfaces are those which provide the capability for objects resident at that interface to be read or modified by an outside task. Required interfaces are those which connect to a provided interface of the same type, allowing the required interface to access specified objects of the provided interface. Thus, data flow travels in both directions, but communication is always initiated from the required interface side.



Figure 3 Task Interface and Class Structure

 Since in the LARS Teleoperation project [1] the details of the class structure is given in this part only the added sections to the existing code will be discussed. Initially, additional to the robotModel class, SnakeModel class is created which holds the static snake robot parameters and robot control structures of the integrated system.

 As known, in CISST multitasking, each task has its own periodic “Run()” function. To explain what is going on the code and what is developed for this integrated system robotTask “Run()” method should be examined. In doing this the data flow in Figure 3 can be better understood.

 In a sample robotTask “Run()” function first the LARS and Snake joints parameters are obtained from the LARS encoders and from the Snake Computer via its own encoders. In updating the snake joint position data and sending the updated Snake joint positions UDP is used as a communication tool between two computers. Then, using the derived forward kinematic equations of the combined system tip current tip position and orientation is calculated. Following this, combined Jacobian of the system is calculated. The details of this calculation will be given in Kinematic Overview part of this report.

 After calculating the combined Jacobian, the next thing is the implementation of the constrained control algorithm in response to an incremental position input command from the 3D Space Mouse. In doing this, the objective function and constraint function matrices are formed. Depending on the mode of operation RCM constraints and Virtual Wall constraints are added to the matrices. Using linear least square solver of the CISST library the optimization problem is solved and as an output the incremental joint positions are obtained. This algorithm output is sent to the Snake Computer FireWire Controller to convert this displacement into a suitable snake actuator encoder values. The same procedure is implemented in LARS side Galil Controller as well.

**Kinematic Overview**