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.

Figure 1

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.

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:

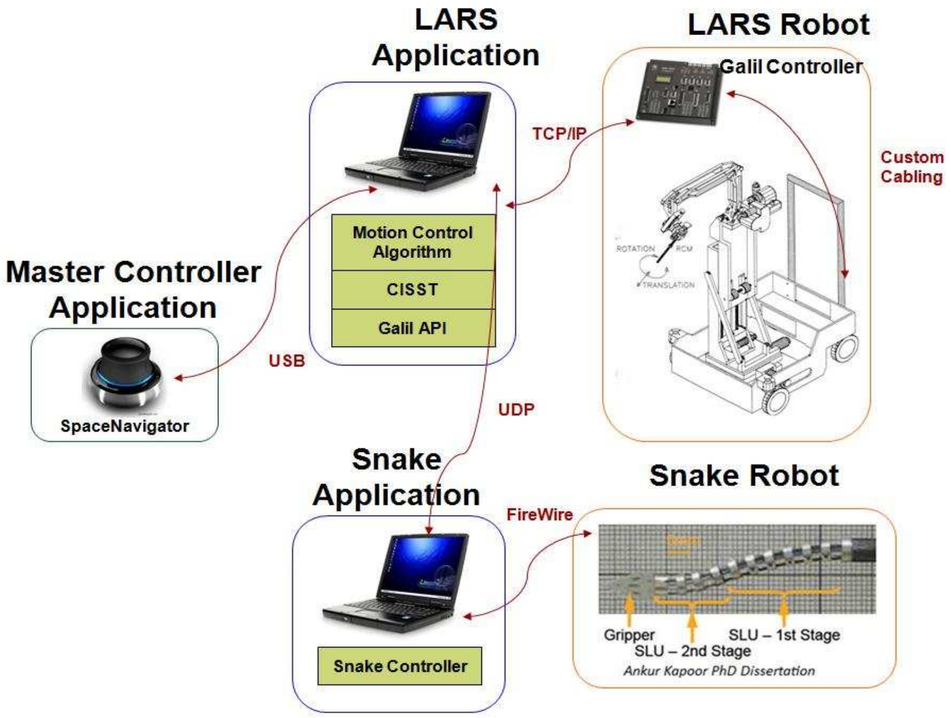

Figure 2 - LARSnake System Overview

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.

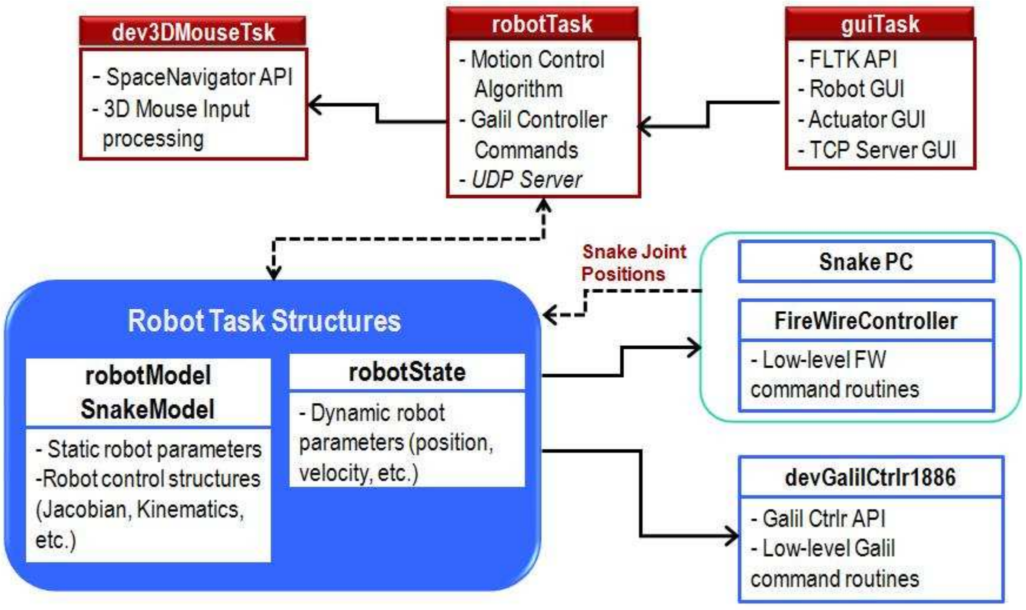
dev3DMouseTsk
- SpaceNavigator API
- 3D Mouse Input processing
robotTask
- Motion Control Algorithm
- Galil Controller Commands
- UDP Server
guiTask
- FLTK API
- Robot GUI
- Actuator GUI
- TCP Server GUI
Snake Joint Positions
Snake PC
FireWireController
- Low-level FW command routines
Robot Task Structures
robotModel SnakeModel
- Static robot parameters
-Robot control structures (Jacobian, Kinematics, etc.)
robotState
- Dynamic robot parameters (position, velocity, etc.)
devGalilCtrlr1886
- Galil Ctrlr API
- Low-level Galil command routines

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.

Kinematics

Forward Kinematics

The first task in controlling the robot is to find the combined forward kinematics map. In the previous studies [1, 2, 4] detailed kinematic analysis of both systems are employed. In this project what needs to be done is to develop a method to combine both forward kinematic equations. In doing this I carried all the snake frame equations into the LARS coordinate frame. The Snake and the LARS coordinate frames can be seen in the below Figure 4 and Figure 5.

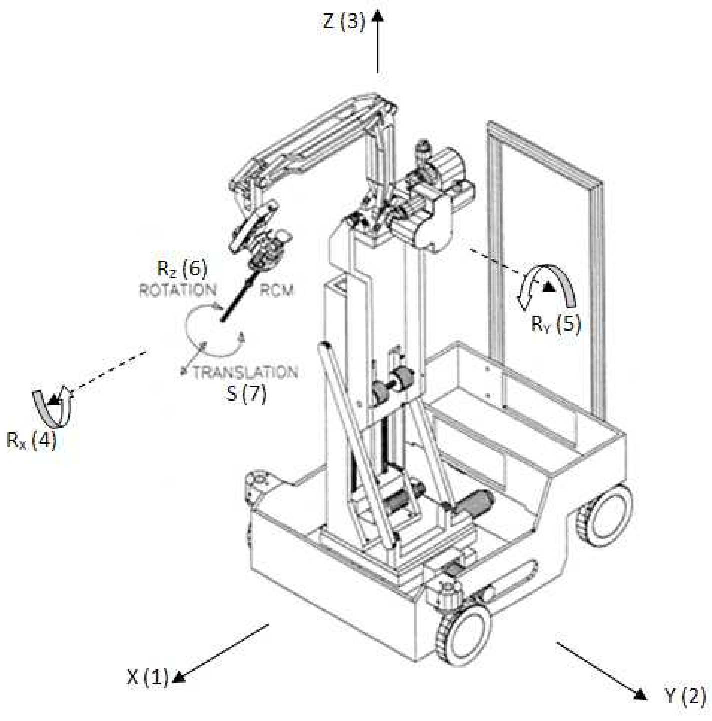

Figure 4 - 7 DOF LARS Robot (taken from robot's design folder)

Physically speaking, Snake Robot is to be mounted on the insertion (s(7)) axis of the LARS robot. A tricky point which must be taken into account is that the direction of z-axis of the LARS robot and the Snake robot. As can be seen the directions are reversed when the robot in Figure 5 is mounted upside down. So the existing Snake forward kinematic equation should be subject to this transformation for which case the 4x4 transformation matrix of the Snake forward kinematic matrices should be multiplied with:

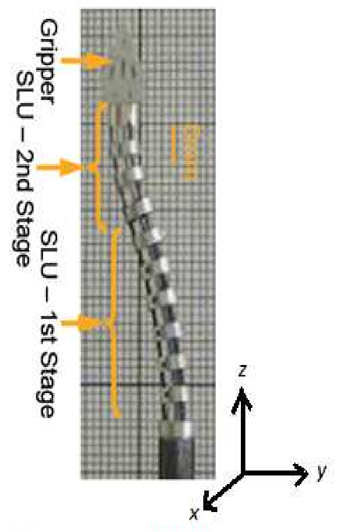

Figure 5 - Snake Robot [2]

$$
f_{base} = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}
$$

which is nothing but a rotation matrix rotated around x-axis by 180 degrees. Therefore, in order to have a fully combined forward kinematic map of the system, expressing matrices vectorial domain, a vector going from the world coordinate frame to the tip of the robot can be expressed as:

$$f_{TIP} = f_{LARS} * f_{base} * f_{snake1} * f_{snake2} \quad ....(1)$$

After this transformation is done on the snake forward kinematics model, the equations become ready to use. Now the next job is the deriving the Jacobian matrix of the combined system.

<u>Jacobian</u>

At the beginning of the semester, I had two distinct Jacobians at hand such that:

Snake Robot: $$\dot{\bar{x}} = J_{snake} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{R}_z \\ \dot{\theta}_1 \\ \dot{\delta}_1 \\ \dot{\theta}_2 \\ \dot{\delta}_1 \end{bmatrix} \quad ...(2)$$ LARS Robot $$\dot{\bar{x}} = J_{LARS} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{R}_x \\ \dot{R}_y \\ \dot{R}_z \\ \dot{s} \end{bmatrix} \quad ...(3)$$

Since snake robot was connected to the Da Vinci surgical system it has a Jacobain of 6x8 and LARS has a Jacobian of 6x7. The aim is to develop a method to combine these two systems such that it will become:

$$\dot{\bar{x}} = J_{tip} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{R}_x \\ \dot{R}_y \\ \dot{s} \\ \dot{R}_z \\ \dot{\theta}_1 \\ \dot{\delta}_1 \\ \dot{\theta}_2 \\ \dot{\delta}_1 \end{bmatrix} \quad ....(4)$$ in which J_{tip} is a 6x11 matrix.

In the Jacobian equations above the left hand side is the velocity of the tooltip both in translational and rotational. However, in medical robots since quickly moving links are not desired the motion is to be pretty slow and smooth. That's why a general assumption is made on these systems such that incremental position and orientation of the end effector is equal to the instantaneous velocities of that links. Similarly, the right hand side of the equation is the incremental positions of each actuator.

At the end of the integration procedure I will have a Jacobian such that:

$$\left[\begin{array}{c:c} \text{LARS PART} & \text{Snake Part} \\ 6\text{x}7 & 6\text{x}4 \end{array}\right] \quad \ldots\ldots.(5)$$

which means that the left part of the Jacobian will be responsible for the LARS joint motion whereas, the right hand side will take care of the Snake motions. This Jacobian should be defined at the tip position of the combined robot which is the tip of the second snake segment. The Jacobian of the LARS is previously defined at the tip position of the LARS robot so through some sort of transformations this LARS Jacobian should be moved to the new tip position. To do this lets first divide the LARS Jacobian into two pieces such that:

$$J_{LARS} = \begin{bmatrix} J_{translational}\ (J_v) \\ J_{rotational}\ (J_w) \end{bmatrix} \quad \ldots\ldots\ldots..(6)$$

When inserting the LARS Jacobian into LARSnake Jacobian, the translational part of the Jacobian becomes:

$$J_{v_{LARS}} = J_v - \left(\left({}^{w}_{b1}R\,{}^{b1}_{g1}p_{b1}\right)^{\wedge} + \left({}^{w}_{g1}R\ {}^{b2}_{g2}p_{b2}\right)^{\wedge}\right)J_w \ \ldots\ldots.(7)$$

where;

${}^{w}_{b1}R$: Rotation matrix from base of the snake frame and LARS frame

${}^{b1}_{g1}p_{b1}$: Vector from gripper-1 to first stage base in base frame

${}^{w}_{g1}R$: Rotation matrix from gripper-1 frame and LARS frame

${}^{b2}_{g2}p_{b2}$: Vector from gripper-3 to second stage base in second base frame

In the above equation (7) "^" refers to a cross product sign.

In addition to the modification done on LARS Jacobian, another modification is needed on Snake Jacobian in which the effect of rotations around x, y and z axis should also be taken into account such that every vector defined in snake frame should be moved to the world coordinate frame taking into account the rotation of the LARS robot. In the derivation of snake Jacobian gripper vectors are being moved to the base frame of the Snake. However, in our combined system the vectors should be moved to the LARS base frame so similar to what I did in deriving the forward kinematics map the vectors ${}^{b1}_{g1}p_{b1}$ and ${}^{b2}_{g2}p_{b2}$ defined above should be multiplied with the below transformation matrix.

$$R_{transformation} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ -\sin(\gamma) & -\cos(\gamma) & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

where β, α and γ refer to the rotation angles around x, y and z axis correspondingly.

Control

The system developed has 7+4=11 degrees of freedom. So the system is already kinematically redundant such that, for 3 translational and 3 rotational unknowns (x, y, z, R_x, R_y, R_z) there are 11 variables to be determined.

The LARSnake robot motion can be classified as:

1) Coarse Motion
2) Fine Motion

In coarse motion, the snake will not move so the system will turn out to be 7 degrees of freedom LARS. So in implementing the constrained optimization algorithm, in developing the objection function matrix the coefficients of the snake actuators should be penalized more such that they should be larger.

In fine motion, it is aimed that x, y, z axis actuators will not move and the system will be an 8 dof system. In this motion RCM mode operation can also be developed as given as an example in the study [3].

In the optimization problem, $||A\Delta q - b||$ is the objection function under the constraints $C\Delta q - d \geq 0$. As noted above the first critical part of the implementation process is the forming the objection "A and b" matrices. The first part of the "A" matrix will be made of the combined Jacobian of the system under concern. Below this, the weights of each actuator will reside diagonally. Depending on the type of the motion described above the penalization constants will differ. Likewise, the first part of the "b" column vector will be 6x1 the desired position and orientation of the end effector. Below it there will be zero.

Now that the objection function is formed, the next job is the formation of the constraint matrices "C" and "d". In snake side of the system, the constraints play a very important role because the capabilities of the snake robot are limited. It can bend around -40, +40 degrees from its equilibrium point so certain constrains are needed in order not converge to a value out of this range. In snake currently it is planned implemented:

1) Snake Joint Positive Limit Constraint
2) Snake Joint Negative Limit Constraint
3) Snake Joint Positive Rate Limit Constraint
4) Snake Joint Negative Rate Limit Constraint
5) Backbone Rate Constraint

Besides, in RCM mode certain constraints at the LARS side of the robot should be put. In doing that, the objection Jacobian should be changed such that the Jacobian should be recalculated at the RCM point. Again in this motion Cartesian axis actuators of the LARS side should be penalized more in order the keep that RCM point at a fixed location in space.

In addition to the RCM mode of operation, a simulation of the Virtual wall is also implemented in this study. As described in [4], a virtual RCM can be implemented on the robot as follows. Assume, for now, that the tool shaft is passing through the virtual RCM (point X0) at the current time step. The robot task then would be to maintain that point on the axis of the tool shaft, i.e. movements of that point in the directions perpendicular to the shaft should be restricted to a small value ε . This way we are restricting that point to move inside a virtual cylinder around the tool shaft. This problem, in its original form, is nonlinear since it involves finding the distance from X0 at the next time step to the fixed virtual RCM

The result of the optimized control with both the virtual wall and virtual RCM constraints can be seen in the below Figure 6.

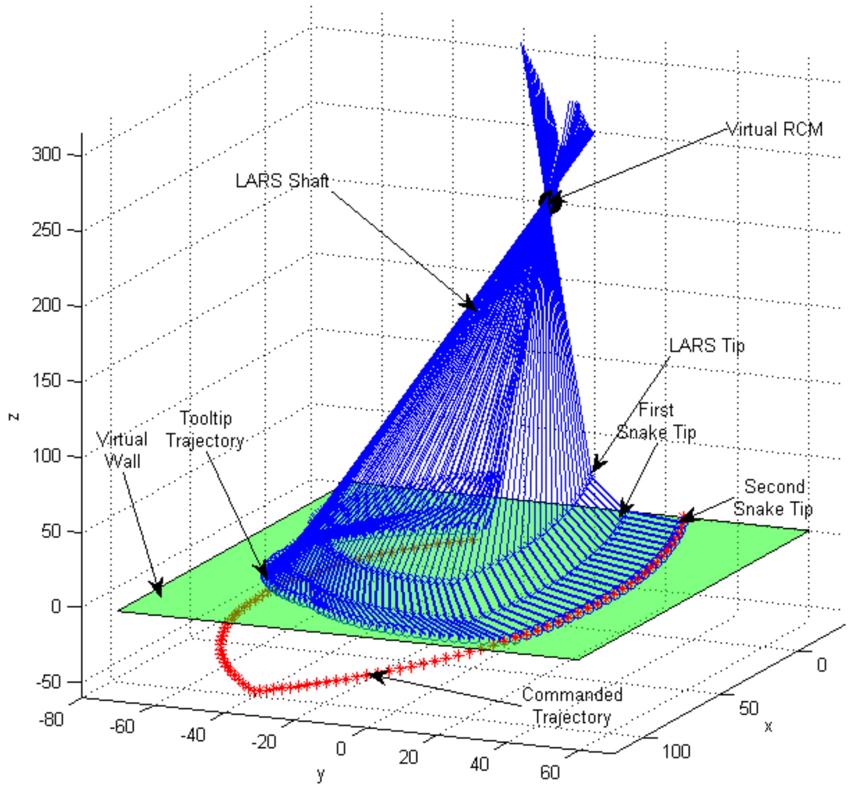

Figure 6 RCM and Virtual Wall Simulation

As can be seen, the robot tip followed the described trajectory successfully. When the virtual wall is introduced the tip remains on the wall while trying to keep the error between the commanded trajectory and second snake tip minimum. From the cone shaped LARS Shaft trajectories, it can be seen that the virtual RCM constraint is implemented successfully.

Colclusion

In this project the integration of LARS and Snake robot is successfully implemented compatible with the CISST software development framework for robot applications. Kinematics equations of the combined systems were derived for the LARSnake robot and various constrained optimization control algorithm was used to teleoperate the robot with the space navigator. At the end, RCM mode operation and virtual wall is simulated in MATLAB.

This project provided a valuable learning experience concerning implementation of robot systems. With the help of this project I am familiarized with JHU's software support for robotic applications, including familiarity with the CISST development framework and the constrained motion control algorithm. I look forward to continuing work on this project to further enhance the robot's capabilities and increase its potential for clinical research.

References

[1] Seth Billings and Ehsan Basafa CIS 2 project on “ Teleoperation of LARS Robot”

[2] Ankur Kapoor PhD Dissertation

[3] Funda, R. Taylor, B. Eldridge, S. Gomory, and K. Gruben, “Constrained Cartesian motion control for tele-operated surgical robots,” IEEE Transactions on Robotics and Automation, vol. 12, pp. 453-466, 1996 “

[4] Ankur Kapoor, Ming Li and Russell H. Taylor, Constrained Control for Surgical Assistant Robots, In IEEE International Conference on Robotics and Automation, 2006

Appendix

Whole project is done on C++ environment and simulations are done on MATLAB. Both codes can be reached through the following link

https://svn.lcsr.jhu.edu/robotorium/trunk/apps/LARSnake/