A Hardware-in-the-loop Simulator for Safety Training in Robotic Surgery

Xiao Li[1], Homa Alemzadeh[2], Daniel Chen[2], Zbigniew Kalbarczyk[2], Ravishankar K. Iyer[2], Thenkurussi Kesavadas[3]

Abstract— **This paper presents a simulation-based safety training simulator for robot assisted surgery. While adverse events occur rarely during training, they could be fatal to the patients if they happen during real surgical procedures and are not handled properly by the surgical team. In this work we propose a hardware-in-the-loop robotic surgery simulator with high fidelity of the robot motion in a simulated environment, which is capable of reproducing adverse events during surgery. The proposed simulator is built upon the Raven-II open source surgical robot, integrated with a simulated surgeon console and a safety hazard injection engine, which automatically injects faults into modules of the robot control software. We simulate representative safety hazards seen in the adverse events, related to da Vinci^{TM} robot, reported to the FDA MAUDE database. A novel haptic feedback strategy is provided to the operator when the underlying dynamics differ from the real robot states.**

I. Introduction

The field of surgical robotics has been rapidly expanding over the last decade [1]. Robot-assisted surgical training is preferred for a variety of minimally invasive medical procedures worldwide. As a technique, simulation based learning and training is gaining acceptance because healthcare professionals improve performance and reduce errors through comprehensive medical care simulation [2] [3]. Simulation can bridge the gap in learning robotic surgery skills without accidentally harming the patient. LapSim Haptic System^{TM} is a laparoscopic surgery simulator which has realistic hardware interface and tactile feedback [4]. This simulator focuses on human-machine interaction but is only used for near-field non-teleoperated surgery training. Other commercial surgical simulators on the market such as Mimic's dV-Trainer^{TM} [5] and Simulated Surgical System's RoSS [6], provide basic motor skills training modules using virtual reality with surgeon console similar to the da Vinci surgical system, to provide life-like simulation and help prepare surgeons. Fig. 1 shows these simulators' profiles. However, a key issue with simulation based surgical training is the lack of safety-critical incident scenarios in simulation-based curricula, which is critical in bringing this form of surgical education to practice. Our previous study of adverse events reported to the U.S. Food and Drug Administration (FDA) Manufacturer and User Facility Device Experience (MAUDE) database, showed that despite significant improvements in robotic surgery technology through the years and broader adoption of the robotic approach, there are ongoing occurrences of safety incidents that negatively impact patients. The number of injury and death events per procedure has stayed relatively constant since 2007, with an average of 83.4 events per 100,000 procedures [7]. The ability of current robotic surgery technology to automatically mitigate the impact of safety-related incidents is not comparable to the situation in other safety critical industries, such as commercial aviation. In such industries, great effort has been spent over the years on improving safety practices by providing comprehensive simulation-based training that includes operation in the presence of safety-critical failures [8]. However, in current robotic surgeon training, the emphasis has been only on improving surgical skills and not on handling safety-critical events and responding to technical problems. Adverse events or machine failures are rarely used as potential scenarios for safety training of surgical teams.

Motivated by the idea of simulating safety hazards [9] [10] during robotic surgery training in order to prepare surgeons for handling safety-critical events, we created a hardware-in-the-loop simulator platform. The objective is to demonstrate the feasibility of using software-based fault injection to simulate realistic safety hazard scenarios in a simulated surgical environment and to provide awareness of the impeding hazards to the operator through haptic force feedback. In this work, we use Raven-II^{TM} [11] surgical robot as the hardware that the operator will be trained with. We believe that the skills are transferable to da Vinci system [12]. We develop a separate robot dynamical model to run simultaneously with the Raven-II robot as the robot's nominal states estimator (fault free run), and a safety hazard injection engine that inserts faults to the robot control software after the system's automatic safety checks are performed. This way the chances the inserted fault will cause a surgeon visible system error or safety hazards is high.

This paper is organized as follows: Section II describes the architecture of our simulator system. Section III reviews the Euler-Lagrange approach of Raven-II robot dynamics modeling and describes the haptic force feedback mechanism. Section IV introduces the safety hazard injection engine used to create adverse events and its integration with the Raven-II software. Section V presents the experimental results and discusses the effectiveness of robotic surgeon training with our new simulator. Concluding remarks are given in Section VI, in which we summarize our results and the future directions of our research.

II. Simulator System Architecture

A. System Overview

We design the simulator system based on the Raven-II surgical robot, an open source platform running on top of

[1]Department of Mechanical Engineering, xiaoli16@illinois.edu
[2]Department of Electrical and Computer Engineering, {alemzad1,dchen8, kalbarcz, rkiyer}@illinois.edu)
[3]Department of Industrial and Enterprise Systems Engineering, kesh@illinois.edu
University of Illinois, Urbana, IL, 61801, USA

Fig. 1. From left to right: LapSim Haptic System, dV-trainer, and RoSS.

Robot Operating System (ROS). To develop a surgical simulator with high fidelity in reproducing adverse events, we include the robot hardware in the simulator's execution loop and integrate it with a safety hazard injection engine [13]. The simulator system architecture is shown in Fig. 2. Raven-II is a tele-operated surgical robot which uses network communication between the local machine on surgeon's console side and the remote Raven computer. The simulator runs on the local machine and performs kinematics, dynamics, and collision detection calculations. Two Phantom Omni devices receive the incremental motion command from the operator, then send the data to both the local machine and the remote machine through UDP/IP. A virtual Raven robot and 3D surgery environment are displayed on the screen of surgeon's console. The graphics are rendered through C++ OpenGL pipeline with a rendering frequency of 30Hz, while other calculations using multiple threads and network sockets are being synchronized and run at 1000Hz, which is the same as the running frequency of Raven's control loop.

The connection between the Surgeon Console and the Raven-II system is two-way network communication in our hardware-in-the-loop simulator shown in Fig. 2. One direction is for transmitting Omni command data from the local machine to the remote Raven system, while the other direction is the reverse, for sending the robot state data (joint positions and velocities) back to the local machine using TCP/IP socket connection for reliability and to make comparison with the dynamics calculation results. The haptic force feedback is provided to the operator if the virtual and real Raven's end-effector trajectories do not match (above a predefined threshold). Since perfect transparency (master device force/torque matching the slave's end-effector force/torque) is not possible, and is especially challenging for teleoperators with significant nonlinear dynamics and no force sensors mounted at the robot end-effector, we utilize haptics feature for safety propose, rather than for surgical palpation. Because the haptic device sensor/actuator asymmetries can cause instability and robustness issues, we apply a spring-damper model with appropriate gains and saturations for feedback force calculation.

To simulate the safety hazards in real surgery, we integrate the robot control software with a safety hazard injection engine that strategically inserts faults into the control software at critical junctures during operation [13]. More explicitly, the injected faults corrupt either the Omni commands or the motor control commands sent to the Raven hardware after the safety checks are done in the software. As a result, unexpected robot motion will generate trajectory errors compared with the underlying model dynamics. Then we show how the operator can gain awareness of the erroneous robot trajectory in presence of faults through haptic force feedback.

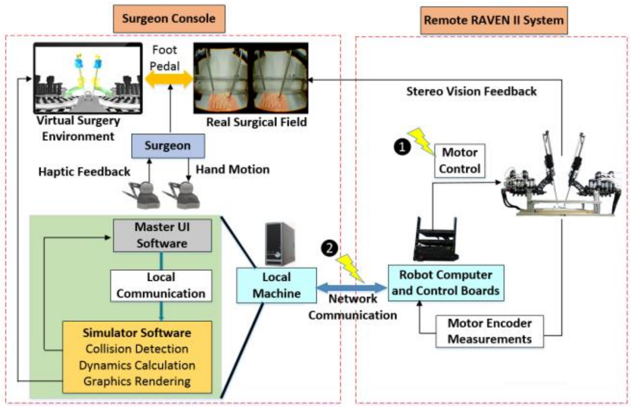

Fig. 2. Hardware-in-the-loop Surgical Simulator Architecture.

B. *Other Capabilities of Hardware-in-the-loop Simulator*

Beyond the capability of basic motor skills training and simulating adverse events, there are additional features in simulator that can help improve the surgeon's performance in real surgery. For example, in some scenarios, it is preferable to let the surgeon do a virtual trial, rather than manipulating the actual robot all the time. One of the important capabilities of our simulator is to allow the user to disengage from the actual robot to do a trial movement in the simulator's virtual environment and see the outcomes of virtual motion. If the outcome is satisfactory, then the actual robot could be re-engaged to track the recorded command trajectory data and move in an autonomous fashion. A foot pedal is placed in the surgeon's side to enable switching between the robot teleoperation mode and pure simulation mode (being disengaged from the robot hardware), and also toggling the view between the real surgical field and virtual environment.

III. Internal Robot Dynamics Modeling and Haptic Force feedback

Raven-II is a 7 degree of freedom (DOF) cable-driven robot, two instrument arms are independent from each other in terms of assembly and controls. The motors for each arm are mounted on the chassis, and pulleys and low-stretch stainless steel cable connect each robot link and motor capstan. Previous work modeled the system dynamics from motor torque input to joint states by adapting an Unscented Kalman Filter, and estimating cable tension parameters for better joint state estimation [14]. In [15] machine learning methods are used to achieve more accurate positioning and kinematic chain calculations while ignoring dynamics. In this work, the robot dynamics is modeled for all 7 DOF, with the last two DOF simplified (two jaws of the grasper), compared to only 3 DOF were modeled in [14]. Our dynamic model is from joint torque to joint states, ignoring the motor dynamics and cable tensions. One reason for doing this is that the cable coupling introduces uncertainties in the model and non-uniformity of cable tension behaviors. In our setting, we tighten the cables as tight as possible before the testing. Another reason is because the system is running at a 1000 Hz frequency, which gives very little margin for heavy computation and introducing even a small time delay will cause system instability. Although the dynamics calculation is done on the Surgeon Console machine rather than the Raven computer, we still do not want to violate the timing constraints in each control loop.

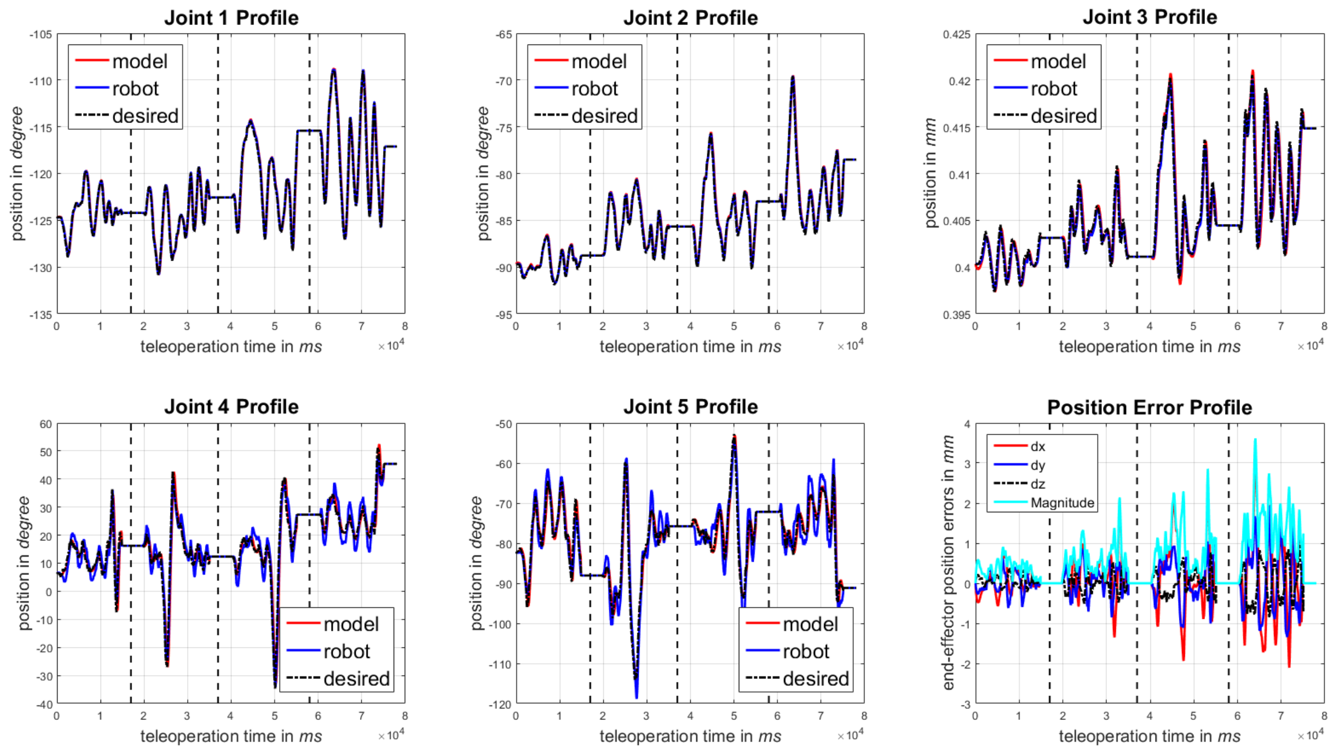

Fig. 3. Comparison of the model and robot running data (up to 5 joints), and end-effector position error of $(2.43 \pm 1.72)mm$

A. E-L Approach of Modelling Open Chain Dynamics

We obtain the mass, inertia tensor, center of mass position for each link by using the robot CAD model files. The system of second order ordinary differential equation are established through Euler-Lagrange approach [16]:

$$M(x)\ddot{x} + C(x,\dot{x}) + G(x) \\ = \tau - \mathrm{diag}(\dot{x})K_v - \mathrm{diag}\big(sign(\dot{x})\big)K_c \tag{1}$$

$$\ddot{\vartheta} = \mu \tag{2}$$

In which x represents the 10 dimensional state vector, C are the Coriolis and centrifugal and G is the gravity term. K_v and K_c stand for the viscous friction and coulomb friction coefficients [14]. The rest two DOF are modeled as simple double integrator model as shown in (2), which represents the two jaws' rotation dynamics.

The joint torque vector τ is the controller output based on desired joint position obtained through inverse kinematics and current joint position. PD controllers are used for joint 1, 2, 4, 5, 6, 7, and PID is used for joint 3, which is the tool insertion translational joint. A set of manually tuned PID gains make this system closely track the desired joint positions while keeping the joint torque/force τ within certain bounds. This means the model is behaving like the robot rather than a system which has low damping and is fast enough to track signal. Fig. 3. shows the trajectories for arbitrary motions provided by the operator (black), the internal dynamic model calculations (red), and the real robot running data(blue) for the first five joints on the left robot arm (the right arm is identical to the left arm in terms of modeling and control), and through forward kinematics chain one can obtain the end-effector position error. In Fig. 3, different portion of the trajectories (separated with dashed lines) correspond to different teleoperation scaling factors, respectively, ranging from 0.05 to 0.2 with spacing interval of 0.05. With larger motion scaling factors, the error also increases, because the modeling error for joints 4 and 5 are more sensitive to the scaling factor.

B. Spring Damper Model for Haptic Force Feedback

In this section, we present the force feedback control mechanism. The use of haptic devices in teleoperated surgical robots has the potential of providing both cutaneous (tactile) and kinesthetic (force) information during exploration or manipulation of an object or environment. To the best of the authors' knowledge, even the latest commercial surgical system (da Vinci Xi) does not have haptic feedback feature. In robotic surgery, haptic feedback is useful in teleoperated palpation [17] [18]. Beyond this application, we expect that haptic feedback also can provide extra but crucial information to the operator about the status of the system when some uncertain events happen and before errors are accumulated to some degree without notice the system is taken to emergency stop. For human perception, our haptic rendering loop in the simulator also runs at 1000Hz, otherwise the user may perceive force discontinuities and a loss in fidelity [19].

We send the published joint states in the Raven computer to the simulator through network. From the dynamic equations we extract the joint velocities from the estimated state vector. We compute the end-effector velocities by using spatial manipulator Jacobian transformation:

$$[v\ \omega]^T = J\dot{\theta} \tag{3}$$

The end-effector position is computed through the forward kinematics chain for both the robot and the model using the joint positions, as shown in (4):

$$p = f(\theta) \tag{4}$$

where f indicates the forward kinematics chain of the robot [20]. Then the haptic force provided to the operator is given by:

$$F = \begin{cases} K_p(p_{model} - p_{robot}) + K_d(v_{model} - v_{robot}) \\ 0.05, \text{ if } \|p_{model} - p_{robot}\| > tol \end{cases} \quad (5)$$

And the force direction applied to the haptic device is given by:

$$d = \frac{p_{model} - p_{robot}}{\|p_{model} - p_{robot}\|} \quad (6)$$

In this setup, if an adverse scenario happens, or the robot moves in an unexpected way, the haptic device will provide haptic cues to the operator. This provides awareness of impeding hazards, enabling the operator to take action or correct the robot behavior based on the internal model of the simulator.

IV. Safety Hazard Injection

Software-implemented fault injection (SWIFI) [21] is commonly used for evaluating the safety and reliability of computing systems. SWIFI validates the effectiveness of fault-tolerance mechanisms by studying the behavior of a system in the presence of faults. Here we use software-based fault-injection techniques to enable evaluation of human operator performance and response to safety hazards in simulation-based training.

Based on our preliminary review of almost 1,500 accident reports on the da Vinci surgical system from the FDA MAUDE database, we identified three common safety hazard scenarios shown in Table 1. We simulate these scenarios by injecting faults into the Raven control software. The possible causes of hazards may include accidental faults in robotic hardware or software or unintentional human operator errors. For each safety hazard, the Table 1 shows the potential causes and impact on patients based on representative examples from the real incidents reported to the MAUDE database. Those patient impacts represent clinical scenarios on which the robotic surgeons should be trained on.

The Safety Hazard Injection Engine consists of customized modules for a) retrieving hazard scenarios, b) generating software fault injection campaign and selecting fault injection strategy, c) conducting fault injection experiments, and d) logging and collecting data in an automated fashion [10]. The *Injection Controller* is responsible for starting, stopping and automating the fault injection campaign. In a normal campaign execution, a Safety Hazard Scenario Library constructed based on the analysis of adverse events is accessed to retrieve the list of desired hazard scenarios. Then causal factors leading to each desired hazard scenario are simulated by selecting the fault injection parameters. Each hazard scenario includes a possible unsafe control action and a list of potential causal factors. An example unsafe control action would be a motor command is provided by the control software when there is a mismatch between the software state and hardware state of the robot. Faulty communication between software and hardware (e.g. through USB) is an example causal factor that might lead to such unsafe control action (see the third example in Table 1). Based on the causal factors involved in each hazard scenario, the analysis of Raven source code, and software/hardware architecture, the *Fault-Injection Strategies* module retrieves information on software functions which can most likely mimic the causal factors leading to the safety hazard, as well as the key variables in those functions and their normal operating ranges. This information is translated to the parameters to be used by the fault injectors for simulating potential causal factors. The fault injection parameters include the location in the software function, the trigger or condition under which the fault should be injected and the target variables to be modified by the injection. Finally, the appropriate software-implemented *Fault Injectors* and the robot software

Safety Hazard Scenario (Outcome)	**Unsafe Control Action Example**	**Possible Causal Factors (Accidental Failures)**	**Raven-II Simulation**		**Patient Impact (Clinical Scenarios for Safety Training) [Example]**
			Target Software Module	**Target Variables**	
System temporarily unavailable **(Recoverable System Error)**	A user command is provided but not followed by the robot	Improper operator actions or console control malfunctions	Network-Layer Thread (*network_layer*)	User-desired -Position -Orientation -Grasper angle -Foot pedal	Restart the system [MAUDE 3293519] Troubleshoot error Contact manufacturer
System permanently unavailable **(Non-Recoverable System Error)**	A motor command is provided by the robot control, but it is not followed by the motors.	Sensor (encoder) failure	Control Thread (*get_USB_packet*)	USB Board -address -returned status	Convert the procedure [MAUDE 2663924] Reschedule [MAUDE 3275500] Report to manufacturer
		Actuator failures	Control Thread (*put_USB_packet*)	USB Board -address -returned status	
Unintended movement of robotic arms **(Sudden Jump)**	A command is provided by the robot control to motors while the calculated next position is at large distance (big jump) from current position.			Commands to robot joints	Puncture of artery [MAUDE 1590517] Bleeding of uterine tube [MAUDE 2120175]

Table 1. Three common safety hazard scenarios, with corresponding examples from real incidents reported to the FDA MAUDE database.

are executed to conduct a fault injection experiment during robot operation. At the end of each injection run, the injection parameters and data are collected for further analysis. For a more detailed description of Safety Hazard Injection Engine refer to [10].

V. EXPERIMENTAL RESULTS AND INSIGHTS

Many of the hazard scenarios shown in Table 1 may cause unexpected instrument movements and sudden jumps. Using the technique introduced in the previous section, we created a safety hazard injection engine that is capable of intentionally injecting unsafe robot state to the robot control software. In this section, we use the safety hazard injection engine to trigger the faulty commands at network layer and software hardware communication layer as in [13]. More specifically, to simulate the resulting safety hazard scenarios, we corrupt the motor commands sent to the robot hardware and the Omni commands sent to the Raven computer, as indicated in Fig. 2.

A. *Fault Injection to Robot Software*

In the experimental setup, we inject periodic faults to i) the first (shoulder) joint of the robot (which has stronger cable in the Raven-II), and ii) the motion command data in network layer transmitting from the local machine to Raven computer, while the simulator receives the original "clean" Omni input. In the second case (in Fig. 2), it is obvious that receiving the corrupted desired state data, the robot will follow the incorrect trajectory and end up deviating from the trajectory expected by the operator. From the fault-free run result as shown in Fig. 3, we set the threshold of triggering the haptic force feedback when the end-effector positions between the model and the robot deviate more than 3mm. In this section, we mainly focus on simulating and analyzing the resulting adverse events in the first case (i.e., injection of periodic faults to the motor commands).

B. *Simulation of Sudden Jump*

In robotic surgery, many reported adverse events can be classified as unexpected joint motion in a small time interval, i.e. sudden jump (third scenario in Table 1). Although the causality can be many to one, we are able to reproduce this kind of adverse events and expose the surgeon during training phase by using our hardware-in-the-loop simulator incorporated with the safety hazard injection engine. We use haptic force feedback to provide information to the operator immediately, so that they can respond to the adverse events as quickly as possible, by emergency actions such as release the foot pedal to disengage the robot and triggering motor breaks (to avoid patient injuries). To simulate robot jump, during the teleoperation running mode, we inject constant motor command (can be zero or nonzero but within the valid range of motor's DAC command) to the shoulder joint at a specified time period. The underlying reason for jump is the accumulation of position errors, because the controller has to generate large torque commands to track the desired position once the robot goes back to the nominal run.

Fig. 4 shows the result of our hardware-in-the-loop simulator running with the fault injector. Every 8-second after pedal down (teleoperation mode), we corrupt the motor command which is used to control the shoulder joint and keep the fault active for 300 cycles (300ms). One can observe the sudden jump behavior happened in joint 1 profile in Fig. 4.

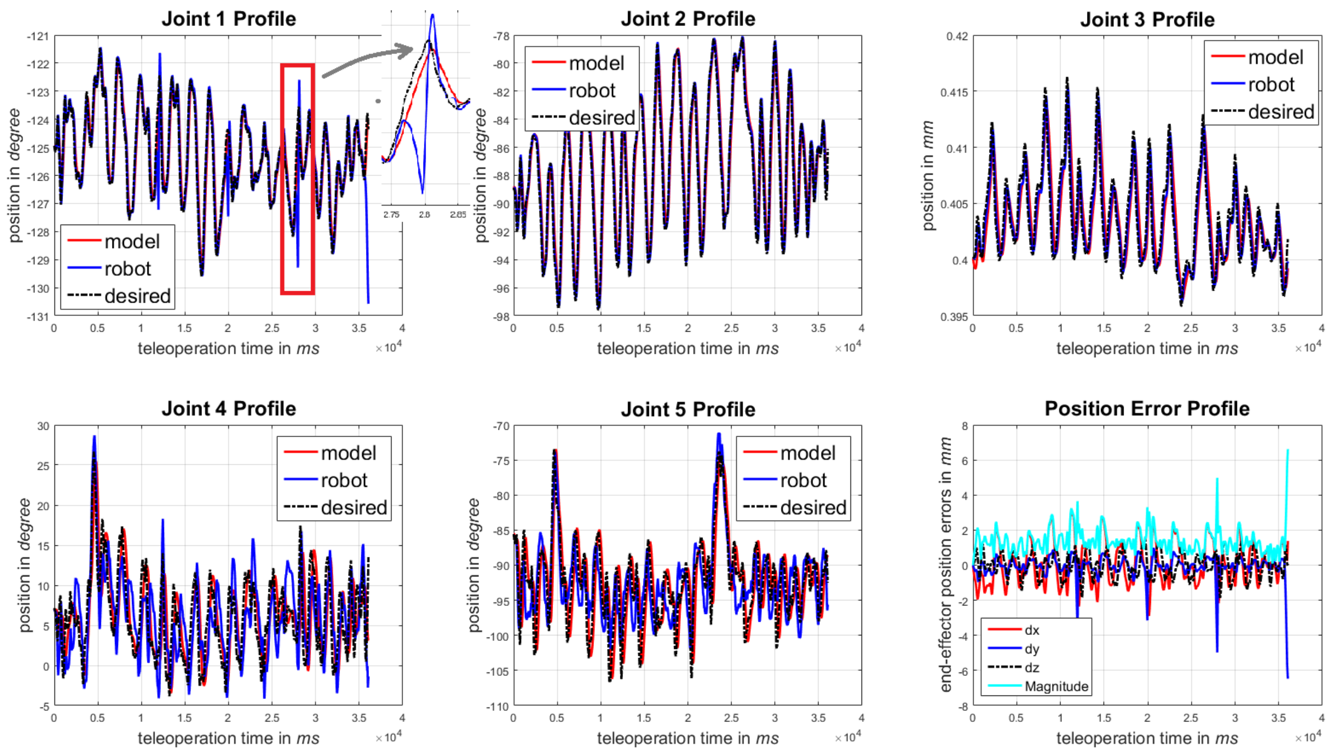

Fig. 4. Robot and model trajectories during fault injection is enabled (with teleoperation scaling factor of 0.1)

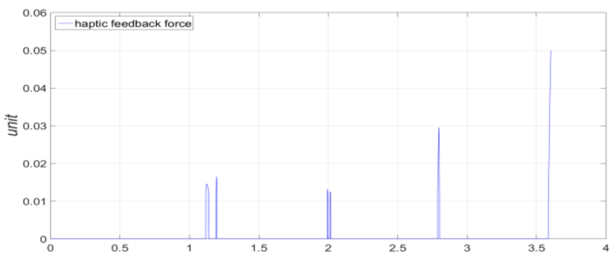

Fig. 5. Haptic force feedback on the Omni device during fault injection

The sudden jumps can happen many times (in this experiment, 4 times), while the operator may not notice since the duration is quite short (a few milliseconds). Such abrupt jumps if only happen a few times during the procedure, they will leave no impression to the operator and he may even think it is his own mistake. However, the sudden movements/jumps may happen due to hardware problems (see Table 1). The robot has the safety mechanisms to monitor the robot status and detect such faults, but in our fault injection experiments we demonstrated the robot can jump frequently without triggering the robot's safety mechanisms [10] [13] (e.g., the robot stopped at the last jump due to the computed motor control is beyond the limit).

Fig. 5 shows the magnitude of the haptic force feedback provided to the Omni device using (5). The results show that we captured the adverse events exactly at the times the fault injection was performed, and provided feedback to the user in time. When a surgeon faces such scenario in real surgery, possible mitigation strategies include slow down the motions or release the pedal to disengage the master and slave and call the technical help in the hospital (see last column of Table 1).

VI. Conclusion and Future Work

With the goal of providing high-fidelity surgical training in simulation, we created a hardware-in-the-loop simulator platform. A full robot dynamic model at joint level enabled us to do the dynamic simulation to mimic the real robot behavior independently. We developed a safety hazard injection engine integrated with the Raven-II robot system and simulator software to reproduce safety hazards happened in real surgery. A haptic force feedback mechanism was designed to provide surgeon an extra modality of information about the robot status when unexpected motion happens. Delivering the safety alarm to the surgeon by haptics is an efficient way of capturing such occurrences but will need additional human factor studies.

We have demonstrated a general framework for robot-assisted surgical simulators for a more robust and resilient robotic surgery. Future work to further enhance the capabilities of our simulator would include: (1) better modeling of robot full dynamics including the uncertainties of cable mechanisms; (2) implementing a richer library of safety hazard scenarios to reproduce them in simulation; and (3) human factor studies with control groups to understand the efficacy of the haptic based safety training in order to verify the practical value of the proposed simulator.

Acknowledgment

This work was partially supported by the National Science Foundation under Award Numbers CNS 13-14891 and CNS 15-45069, and a grant through the JUMP-ARCHES (Applied Research for Community Health through Engineering and Simulation) program for addressing safety and reliability of surgical robots. This project was carried out at the Health Care Engineering Systems Center at Illinois.

References

[1] G. Spinoglio, Ed., Robotic Surgery: Current Applications and New Trends, Springer, 2015.

[2] M. Passiment, H. Sacks and G. Huang, "Medical Simulation in Medical Education: Results of an AAMC Survey," 2011.

[3] Y. Okuda, E. O. Bryson and e. al., "The utility of simulation in medical education: what is the evidence?," *Mount Sinai Journal of Medicine,* vol. 76, no. 4, pp. 330-343, 2009.

[4] "www.surgical-science.com/portfolio/haptic-system/," [Online].

[5] "www.mimicsimulation.com/products/dv-trainer/," [Online].

[6] "www.simulatedsurgicals.com/ross.html," [Online].

[7] H. Alemzadeh, J. Raman, N. Leveson, Z. Kalbarcyzk and R. K. Iyer, "Adverse events in robotic Surgery: A retrospective study of 14 years of FDA data," *PLOS ONE,* vol. 11, no. 4, pp. 1-20, 2014.

[8] F. F. Bilotta, S. M. Werner, S. D. Bergese and G. Rosa, "Impact and Implementation of Simulation-Based Training for Safety," *The Scientific World,* vol. 2013, 2013.

[9] H. Alemzadeh, D. Chen, Z. Kalbarczyk, R. K. Iyer, X. Li, T. Kesavadas and J. Raman, "A Software Framework for Simulation of Safety Hazards in Robotic Surgical Systems," in *Special Issue on Medical Cyber Physical Systems Workshop* , 2015.

[10] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, J. Raman, N. Leveson and R. K. Iyer, "Systems-theoretic Safety Assessment of Telerobotic Surgical Systems," in *In the 34th International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, 2015.

[11] "applieddexterity.com/," [Online].

[12] "Raven surgical robot training in preparation for da Vinci," in *Medicine Meets Virtual Reality 21*, 2014, pp. 135-141.

[13] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk and R. K. Iyer, "Targeted Attacks on Teleoperated Surgical Robots: Dynamic Model-based Detection and Mitigation," in *The 46 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* , 2016.

[14] M. Haghighipanah, Y. Li, M. Miyasaka and B. Hannaford, "Improving position precision of a servo-controlled elastic cable driven surgical robot using Unscented Kalman Filter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[15] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, P. Abbeel and K. Goldberg, "Learning accurate kinematic control of cable-driven surgical robots using data cleaning and Gaussian Process Regression," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2014.

[16] R. M. Murray, Z. Li and S. S. Sastry, A Mathematical Introduction to Robotic Manipulation, 1994.

[17] L. N. Verner and A. M. Okamura, "Force & torque feedback vs force only feedback," in *EuroHaptics conference*, 2009.

[18] J. C. Gwilliam, M. Mahvash, B. Vagvolgyi, A. Vacharat, D. D. Yuh and A. M. Okamura, "Effects of haptic and graphical force feedback on teleoperated palpation," in *IEEE International Conference on Robotics and Automation*, 2009.

[19] OpenHaptics Toolkit Programmers Guide.

[20] "Kinematic Analysis of the Raven-II Research Surgical Robot Platform (REV: 9-Mar-2015)".

[21] M.-C. Hsueh, T. K. Tsai and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75 - 82, 1997.