A Computational Framework for Complementary Situational Awareness (CSA) in Surgical Assistant Robots

Preetham Chalasani, Anton Deguet, Peter Kazanzides and Russell H. Taylor
Laboratory for Computational Sensing and Robotics
Johns Hopkins University
Baltimore, Maryland 21218
Email: {pchalas1, anton.deguet, pkaz, rht}@jhu.edu

***Abstract*—Robotic surgical systems have contributed greatly to the advancement of minimally invasive surgery (MIS). More specifically, telesurgical robots have provided enhanced dexterity to surgeons performing MIS procedures. However, current robotic teleoperated systems have only limited situational awareness of the patient anatomy and surgical environment that would typically be available to a surgeon in an open surgery. Although the endoscopic view enhances the visualization of the anatomy, perceptual understanding of the environment and anatomy is still lacking due to the absence of sensory feedback. To address these limitations, we present an algorithmic software framework to provide Complementary Situational Awareness (CSA) in a surgical assistant. This framework aims at improving the human-robot relationship by providing elaborate guidance and sensory feedback capabilities for the surgeon in complex MIS procedures. Unlike traditional teleoperation, this framework enables the user to telemanipulate the situational model in a virtual environment and uses that information to command the slave robot with appropriate admittance gains and environmental constraints. Simultaneously, the situational model is updated based on interaction of the slave robot with the task space environment. We provide various high-level and mid-level components to provide CSA and illustrate the necessary capabilities required for any robotic platform to readily incorporate CSA. We also demonstrate the use of our framework for constrained model-mediated teleoperation using the open-source da Vinci Research Kit (dVRK) hardware.**

***Keywords*-software framework; dVRK; online estimation; teleoperation;**

I. Introduction

Teleoperation is the most common human-in-the-loop system, where, in the most basic form, the slave robot follows a motion command given by the master device operated by a human. Due to uncertainty in the environment, performing complex manipulation tasks using basic teleoperation can be risky and is completely dependent on the surgeon's visual perception of the environment. In some cases, it is impractical or extremely difficult to visualize the anatomical features. Our proposed framework aims at dynamically providing information of the situational model, making sure the surgeon is aware of the constantly evolving target anatomy. The whole idea of CSA can be summarized in two fundamental concepts; *a) manipulation is mediated by a situational model, and b) the situational model is updated based on information from manipulation.* Our software platform is implemented as a component based framework with various high-level and mid-level components. Figure 1a shows the conventional surgical workflow in an operational theatre and Figure 1b shows the surgical workflow with the addition of CSA.

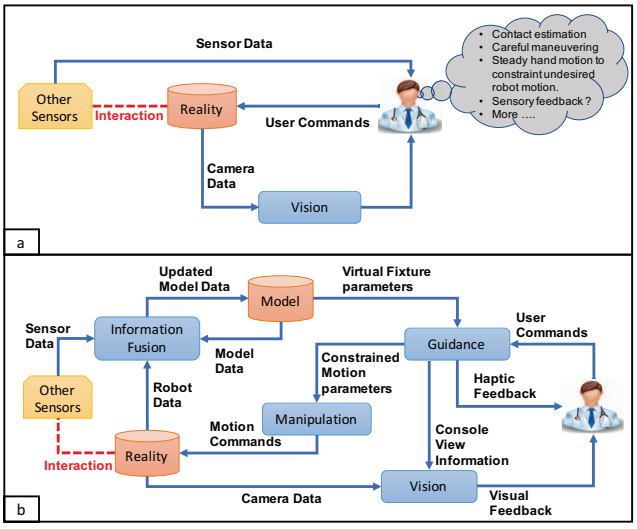

Fig. 1. a) Conventional surgical workflow, b) Surgical workflow with CSA

To provide CSA, any computational framework must incorporate many machine capabilities, including sensor information fusion, enforcing operational constraints for human-robot task execution, providing sensory feedback of the situational model environment, and updating an *a priori* model based on the environment. Further, these functions must be executed simultaneously at interactive, near-to-video frame rates.

There is considerable prior art describing methods for implementing these individual capabilities. Sotiras et al. [1] present a review of various deformable registration methods dealing with environments that deform relative to *a priori* models. In more recent work ([2], [3]), global deformation of a model is addressed.

A number of researchers (*e.g.* [4–8]) have developed finger-like tactile and force sensors to provide information about tool-to-tissue interaction forces. Mahvash et al. [9] developed a

control system for the da Vinci surgical system [10], which provides force feedback with a position-position controller with friction and inertia compensation. Using this system they provided some results on stiffness estimation of a tissue model, based on discrete palpation. More recently, Xu and Simaan have developed methods for estimating the force/moment acting at the tip of continuum robots by using measurements of joint-level actuation forces and extrinsic information regarding the type of interaction with the environment [11], [12]. These methods have been used in [13] to enable force-controlled exploration of flexible anatomy in a manner that benefits from the palpation methods presented in this research.

Mitra and Niemeyer [14] introduced the concept of model-mediated telemanipulation. Based on this approach, Xia et al. [15] demonstrated model-based telemanipulation for satellite servicing using hybrid force/motion control to accommodate environment mismatch with the slave robot. Force controlled exploration has also been examined previously as a means of gathering information for registration and updating a pre-operative model [16]. Constrained Kalman filtering was employed to obtain the rigid registration of the model using contact and estimated stiffness information.

Even though these challenges have been tackled individually, existing approaches have limitations when implemented together in a real-time interactive environment and require a system infrastructure that is non-trivial to implement. Further, when the system has to perform interactively in response to new incoming data, there are additional problems that arise, such as registration and update of the situational model.

Most prior work on image-guided robotic assistance is focused on registering intra-operative to pre-operative images. Global deformation methods such as Coherent Point Drift [2] are also inefficient when new sample points are provided for registration in an incremental manner. Our framework uses an incremental global deformation technique to update the situational model. To estimate surface information, researchers have used discrete probing strategies [17], which can be extremely time consuming and wasteful. The CSA framework supports continuous palpation strategies to dynamically estimate shape and stiffness information during exploration. Current model-based telemanipulation systems lack an efficient way of using this exploration data to update the deformable situational model. Haptic cues and task-specific virtual fixtures can also be highly unintuitive because they are dependent on the constantly evolving situational model. In our own prior work, we have reported concurrent offline estimation of surface shape and stiffness [18] and model-mediated teleoperation using virtual fixtures [19].

This paper presents a component based framework that enables us to address these challenges together. In particular, our computational framework facilitates interactive, online updates of the situational model using information from multiple sources while concurrently updating task-based virtual fixtures based on information obtained during manipulation. Further, our framework dynamically corrects for discrepancies between the situational model and target anatomy.

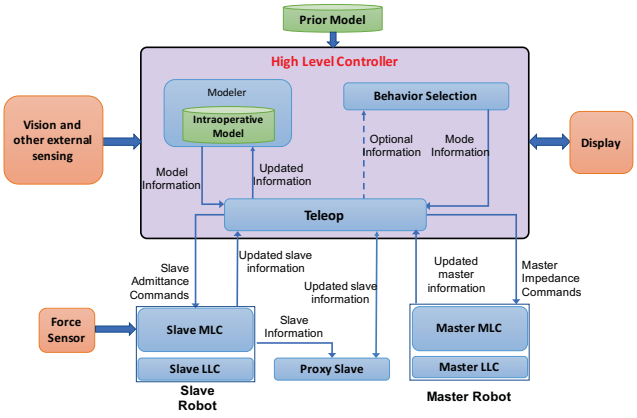

Fig. 2. Various components in the CSA framework.

.

II. Software Framework

CSA is implemented as a component based framework, using the open source *cisst* libraries, developed at Johns Hopkins University ([20], [21]), with support for the Robot Operating System (ROS) [22]. Figure 2 shows the proposed framework and various component interactions to facilitate CSA. *Teleop* is the central high level component managing communication between various components of the framework. The *Master mid level controller* (MLC) sends position to the Teleop; the *Slave MLC* receives these position commands from the Teleop and executes the motion. Each component in this framework is responsible for a particular task and together they provide situational awareness for the surgeon. This computational framework can be incorporated into any robotic platform that can provide the following capabilities: a) Master LLC can be commanded in torque and position control, b) Slave LLC can be commanded in position control and c) Means to report interaction forces.

The following subsections give a brief description of various components and their significance to provide CSA in surgical assistant robots.

A. Master MLC

The Master MLC is implemented as a torque controller. This allows the user to provide external joint torques computed from various control goals. The component is responsible for adding all the necessary joint torques and sending them to the Master LLC which communicates with the master hardware to perform joint-level servo control. One such goal is compliance control, which requires the user to to provide compliance gains to the master MLC. Using these gain parameters, the compliance wrench is computed based on pseudocode described in Algorithm 1. Here the compliance frame $F_c = [R_c, p_c]$ defined in the master base frame is used to calculate the desired wrench $[f, t]$. The position stiffness gains, $\vec{k}^{(+)}, \vec{k}^{(-)}$, position damping gains $\vec{b}^{(+)}, \vec{b}^{(-)}$ and force bias terms $\vec{g}^{(+)}, \vec{g}^{(-)}$ are used to calculate the desired force. Similarly, torque bias terms $\vec{\tau}^{(+)}, \vec{\tau}^{(-)}$ and orientation stiffness gains $\vec{k}_o^{(+)}, \vec{k}_o^{(-)}$ are used

for computing the desired torque. This compliance control is used to provide force feedback to the user on the master handle, which is helpful for user guidance.

The user also has a provision for sending the compliance wrench directly to the Master MLC. This allows for the system to compute the compliance wrench externally and send it to the Master MLC for haptic rendering. As shown in Fig. 3, the user can either specify the external compliance wrench (f_e, t_e) or provide compliance gains for the system to compute the compliance wrench (f, t). The compliance torque τ_c and torque from gravity compensation (τ_{gc}) are added to get the total desired torque.

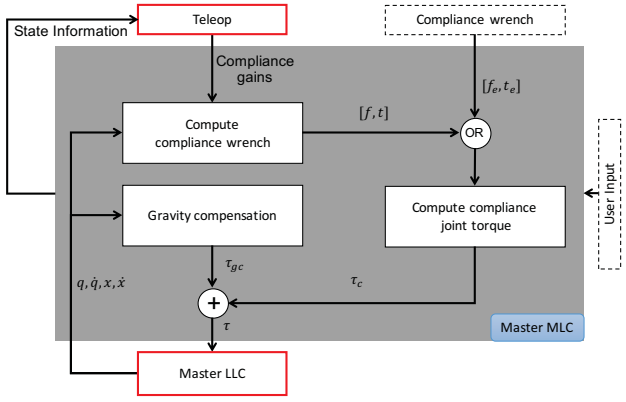

Fig. 3. Master MLC: q-joint position, $\dot{q}$-joint velocity, x-cartesian position, $\dot{x}$-cartesian velocity, $[f_e, t_e]$-compliance wrench from external source, $[f, t]$-wrench computed based on compliance algorithm, τ_{gc}-joint torque from gravity compensation, τ_c-joint torque from compliance control and τ-total joint torque sent to Master LLC

Algorithm 1 Compliance wrench

Output: Wrench $[f, t]$

$\vec{\epsilon} = F_c^{-1}\vec{p} = R_c^{-1}(\vec{p} - \vec{p_c})$ ▷ Position Error

$\vec{v} = R_c^{-1}\dot{p}$ ▷ Velocity on compliance frame

for $i \in \{x, y, z\}$ **do**

  if $\vec{\epsilon}_i \leq 0$ **then**

    $\vec{g}_i = \vec{g}_i^{(-)} + \vec{k}_i^{(-)}\vec{\epsilon}_i + \vec{b}_i^{(-)}\vec{v}_i$

  else

    $\vec{g}_i = \vec{g}_i^{(+)} + \vec{k}_i^{(+)}\vec{\epsilon}_i + \vec{b}_i^{(+)}\vec{v}_i$

  end if

end for

$\vec{f} = R_c\vec{g}$ ▷ Desired force

$\vec{\theta} = Rodriguez(\triangle R = R_c^{-1}R)$ ▷ Orientation Error

for $i \in \{x, y, z\}$ **do**

  if $\vec{\theta}_i \leq 0$ **then**

    $\vec{\tau}_i = \vec{\tau}_i^{(-)} + \vec{k}_{oi}^{(-)}\vec{\theta}_i$

  else

    $\vec{\tau}_i = \vec{\tau}_i^{(+)} + \vec{k}_{oi}^{(+)}\vec{\theta}_i$

  end if

end for

$\vec{t} = R_c\vec{\tau}$ ▷ Desired torque

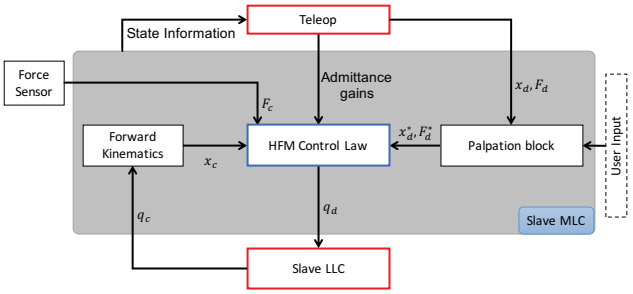

Fig. 4. Slave MLC: q_c-current joint state, q_d-desired joint state, x_c-current cartesian state, x_d-desired cartesian state, F_c-sensed force, F_d-desired limiting force and $\{x_d^*, F_d^*\}$-updated desired position and force based on user specified palpation primitive.

B. Slave MLC

The Slave MLC is responsible for sending motion commands to the slave low level controller (LLC) which communicates with the slave hardware. The Slave MLC is implemented as an admittance type controller while the slave robot follows a hybrid force/motion (HFM) control law. As the name suggests, the control is based on position and force, thus a force sensing component is also connected to the Slave MLC to retrieve the interaction forces with the anatomy. This enables for better perception of the patient anatomy and surgical environment. Figure 4 shows the internal communication of the Slave MLC. HFM control is implemented as an optimization problem and the setup for obtaining the desired incremental joint motion is stated below,

$$\delta q = min \|J\delta q - \delta x\| \tag{1}$$

such that,

$$\delta x = K_a K_g (F_c - F_d)\delta t + K_p(x_d - x_c)$$
$$v_l \leq \frac{\delta q}{\delta t} \leq v_u$$

where, J represents the body jacobian of the slave robot, δt is the sampling rate of the component in seconds and δx and δq represent incremental cartesian and joint position, respectively. K_p and K_a are motion force projection matrices as described in ([23], [24]). K_a projects the motion from the force controller in the direction normal to the surface. Similarly, K_p projects motion from the position controller in the direction perpendicular to the surface normal. F_c denotes the contact force obtained from the force sensor and F_d denotes the desired force the slave should maintain with the anatomy. K_g is the admittance gain matrix for the force motion. x_c is the current cartesian position and x_d is the commanded/desired cartesian position. Inclusion of the constrained optimizer in the Slave MLC also allows the user to incorporate task-space virtual fixtures.

The Slave MLC also has a provision for choosing different motion primitives for continuous palpation. Currently, two motion primitives are supported:

1) Sinusoidal force reference: Motion is achieved when following a sinusoidal force reference. The desired force

F_d in (1) is replaced by F_d^*,

$$F_d^* = F_d + A * Sin(2\pi\omega t) \quad (2)$$

where A, ω, t are parameters of the sine motion.

2) Sinusoidal motion reference: Motion is achieved by applying a sinusoidal motion reference in the direction of the sensed force vector. Desired motion x_d in (1) is replaced by x_d^*,

$$x_d^* = x_d + \hat{n}A * Sin(2\pi\omega t) \quad (3)$$

where A, ω, t are parameters of the sine motion and $\hat{n} = \frac{F_c}{||F_c||}$ is the direction of the sensed force vector.

Figure 5 properly depicts the motions of these primitives.

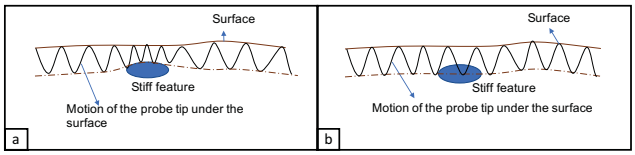

Fig. 5. Palpation motion primitives : a) Sinusoidal force reference, b) Sinusoidal motion reference

C. Proxy Slave

As stated before, CSA focuses on interacting with the situational model and uses information from sensing and interaction with the modeled environment to update the model. The situational model in the virtual environment is registered to a simulated slave robot, called the *Proxy Slave*. This component obtains position commands from the Teleop component and executes the motion based on the position controller only. The minimization problem is similar to that of eq. 1:

$$\delta q = min\|J\delta q - \delta x\|$$

such that,

$$\delta x = (x_d - x_c)$$
$$v_l \leq \frac{\delta q}{\delta t} \leq v_u$$

The user teleoperates the Proxy slave and based on the interaction with the situational model, Teleop sends impedance commands to the Master MLC and admittance commands to the slave MLC. The advantage of the proxy slave is that it provides the state information of a slave robot with an ideal position control. This allows the Teleop component to use the contact information of the proxy slave with the situational model for haptic rendering on the master side. Additionally, Teleop uses this information to send admittance commands to the slave side for better control.

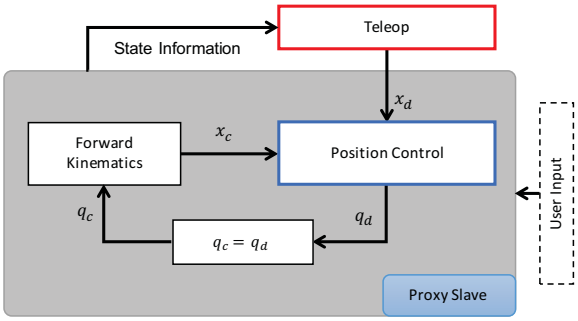

Fig. 6. Proxy Slave : q_c-current joint state, q_d-desired joint state, x_c-current cartesian state, x_d-desired cartesian state

D. Teleop Component

Teleop is a high-level component which maintains various state information and is responsible for managing communications among various components as shown in Fig. 8. It retains the state information of the master robot, slave robot and the proxy slave robot. Since the user is teleoperating the proxy slave, the virtual environment will eventually drift from reality when there is an interaction with the anatomy. This is due to the additional force controller in the Slave MLC. However, the system corrects this mismatch between the virtual and real environment using the contact information of both the slave robots with their respective environments. A sample mismatch scenario is demonstrated in Fig. 7. At time step t_0 the slave robot and the proxy slave robot are at the exact same location in the real and the virtual environment, respectively. Based on master movement, at time step t_1, both receive an incremental move (δx) command from the Teleop. At time step t_2, the Slave robot, obeying the HFM control, reaches the surface of the model and servos at the point of contact maintaining minimal contact force. However, Proxy Slave, obeying position control, will cross the surface in the virtual environment, creating a mismatch between the joint state of the Slave robot and the Proxy Slave.

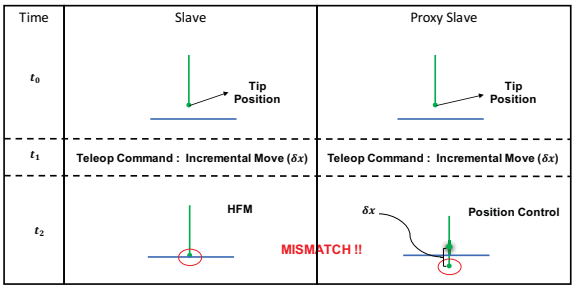

Fig. 7. Sample mismatch scenario

Following are different teleop states (Fig. 9) that are implemented to handle various mismatch scenarios:

- **`PROXY_CONTACT`**: This is the state of Teleop when the proxy slave is in contact with the situational model but the slave robot is not in contact with the anatomy.

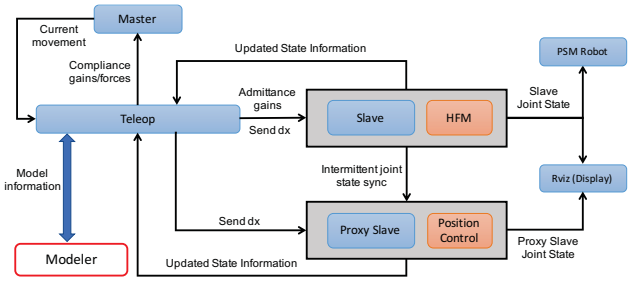

Fig. 8. Teleop information flow

Correction: Compliance forces are sent to the Master MLC for haptic rendering. Desired force (F_d) is calculated and sent to the Slave MLC according to the following equation:

$$F_d = k_d \vec{\epsilon}_s \tag{4}$$

where $\vec{\epsilon}_s$ is a vector defining the depth of the Proxy Slave's tip inside the surface and k_d is a scaling factor resembling stiffness of the tissue. This correction will eventually change the teleop state to **`FULL_CONTACT`** or **`NO_CONTACT`**.

- **`SLAVE_CONTACT`**: This is the state of Teleop when the proxy slave is not in contact with the situational model but the slave robot is in contact with the anatomy.
  Correction: Send zero desired force to the Slave MLC and set the projection matrices in the HFM law to be identity. This will enable position control in the force regulating direction and eventually the teleop state will switch to **`FULL_CONTACT`** or **`NO_CONTACT`**
- **`FULL_CONTACT`**: This is the state of Teleop when the proxy slave is in contact with the situational model and the slave robot is also in contact with the anatomy.
  Correction: Compliance forces are sent to the Master MLC and admittance gains are sent to the slave MLC. Additionally, the direction of the desired force is calculated based on the direction of the sensed contact force. This will make sure that, if palpation is enabled, the force reference direction is close to the surface normal direction.
- **`NO_CONTACT`**: This is the state of Teleop when the proxy slave and the slave robot both are not in contact with the situational model and the anatomy, respectively.
  Correction: This is a special case where both are not in contact but there still might be a mismatch due to the previous state of Teleop:
  1) If the previous state was **`NO_CONTACT`** or **`FULL_CONTACT`**, then there is no mismatch.
  2) If the previous state was **`PROXY_CONTACT`** or **`SLAVE_CONTACT`**, then the proxy slave's state information is synced with the slave robot.

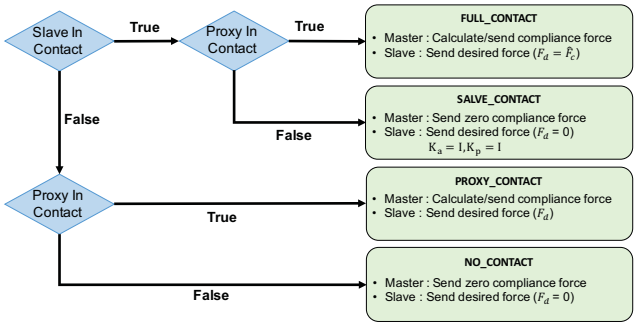

Fig. 9. Different Teleop states to correct mismatch

The Teleop component also retains the current model information. Based on the contact information of the proxy slave with the situational model, compliance forces are calculated using algorithm 1 and sent to the Master MLC.

This design allows the system to dynamically detect and correct the mismatch between the virtual environment and the reality. Simultaneously, the system also provides smooth haptic rendering when interacting with the situational model.

E. Modeler

The Modeler is responsible for providing the Teleop component with the latest model information based on the interaction between the slave robot and the anatomy. Additionally, it performs an optimization step to provide the optimum location for exploration to maximize information gain. This task is divided into three sub processes:

1) **GP Process**: This process is responsible for providing surface and stiffness information based on continuous palpation. The method is based on Gaussian Process (GP) and our work on offline estimation is described in [18]. However, for the framework to execute in real-time the estimation has to be done incrementally. Our new approach is based on learning the local force model around the palpating region using GP. This model is then used to incrementally estimate the local stiffness and shape of the anatomy while it is being palpated. Spatial data structures, specifically hash grids, are used to store position and force information for efficient data storage and retrieval. This allows for low-latency estimation of local surface information around the region of palpation.
2) **Registration process**: This process is responsible for providing the latest model information by registering the current model with the information provided by the GP component. We implemented an online deformable registration technique based on work done by Billings et al. [25]. We use a grid-based deformation technique where the undeformed model mesh is enclosed with sparse grid vertices. The deformation field is then optimized over the grid vertices rather than the model vertices for faster computation.
3) **Trajectory optimizer**: We have implemented a preliminary version of a trajectory optimizer formulated as a stochastic optimization over trajectories parameterized using a finite dimensional vector (FDV). This process is responsible for providing the next optimum location on

the model based on a user specified cost function. We presented some initial results of this method in [18], which is based on an upper confidence bound (UCB) on the stiffness estimate. We considered two different cost functions for a given FDV: one minimizing the variance of stiffness and thus seeking to improve overall model quality and the other seeking maximum stiffness. Optimization is performed using a cross-entropy method as detailed in [26], [27].

Our colleagues at Carnegie Mellon University are further exploring different strategies for path optimization that can be easily incorporated in the CSA framework ([28–30]).

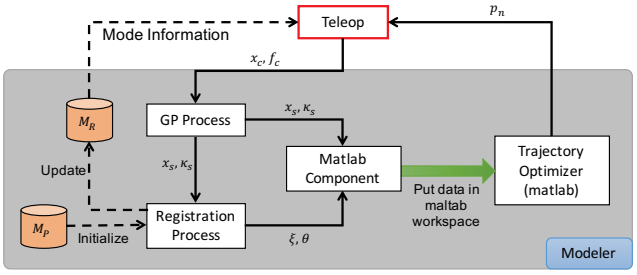

Fig. 10. Modeler : $[x_c, f_c]$-current robot tip position and sensed force; $[x_s, \kappa_s]$- estimated surface point and stiffness; $[\xi, \theta]$- shape and registration parameters; M_R-situational model in robot space and M_P-preoperative model

Figure 10 shows detailed information flow between various processes of the modeler component. The GP component continuously receives position and force information (x_c, f_c) from the Teleop component and stores it in a spatial data structure. In this process, surface shape and stiffness (x_s, κ_s) are recursively estimated in the neighborhood of the current end-effector location. This information is then passed to the registration process, which computes an optimized set of shape and registration parameters (ξ, θ). Using these parameters and the preoperative model (M_P), the situational model is updated (M_R). The Matlab component is responsible for retrieving the latest surface and shape information $(x_s, \kappa_s, \xi, \theta)$ and passing it to the trajectory optimizer. Based on this information, a new location (p_n) is calculated and sent to Teleop for further exploration.

This task division of modeler component is useful in expanding the system for future inclusion of different surface estimation, registration and optimization techniques.

III. System Integration

The high and mid-level components provided by our framework can be configured with any low-level controllers that provide the necessary interfaces. In this section, we demonstrate the CSA integration with the open source da Vinci Research Kit (dVRK) [31], [32], along with an example application.

A. Build, Dependency and Support

The CSA framework is compiled against the open source *cisst* libraries, developed at Johns Hopkins University ([20], [21]). The build system is a catkin-based solution as described in [32], where all the CSA packages depending on *cisst* are built as catkin packages. This allows the users to download the CSA code and compile using ROS catkin tools. This makes ROS one of the core dependencies of the CSA framework. Additional external dependencies include Eigen (https://eigen.tuxfamily.org), PCL (wiki.ros.org/pcl), dlib (www.dlib.net) and WildMagic (www.geometrictools.com). All these dependencies already exist as catkin packages or are included in the CSA code as catkin packages created from their latest versions.

We also provide support for evaluating MATLAB code in C++ using the MATLAB Engine API. This allows the users to have their code in the MATLAB workspace and call MATLAB functions with arguments from C++. The API also allows the system to directly exchange data between MATLAB and C++, rather than communicating over a ROS channel. The user is also provided with an interactive shell to type in MATLAB commands in the C++ workspace. This allows the users to interactively view their data while the application is running.

B. Integration with dVRK

The dVRK supports custom configuration of arms using Javascript Object Notation (JSON) files, thus facilitating integration of CSA in the dVRK framework. The Master MLC, Slave MLC and Teleop components of the dVRK are replaced with the CSA components via JSON configuration. This allows the CSA components to maintain communication with the dVRK low-level controllers, since CSA is also a component based architecture developed using *cisst*. We used this integration for various applications in the past ([18], [19]) and below is a demonstration of one such setup. Although this integration takes advantage of the *cisst* framework, this is not a requirement because integration can also be performed via ROS or the UDP interface described in Section III-C1.

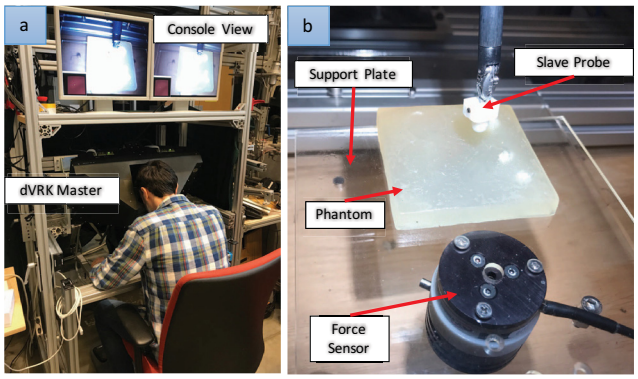

Fig. 11. a) User teleoperating using the dVRK master device; b) Slave side setup.

To demonstrate the system, we created a silicone phantom with an embedded stiff feature resembling a tumor at an unknown location. An ATI Mini-25 force-torque sensor (ATI Industrial Automation, Apex, NC, USA) was used to measure

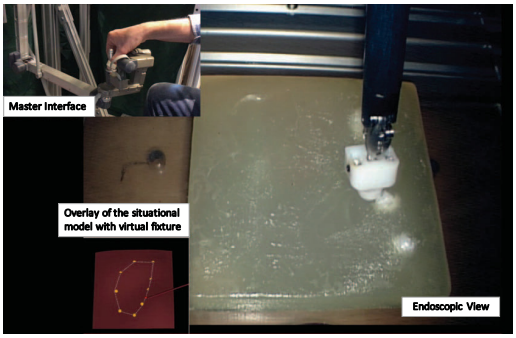

Fig. 12. Endoscopic view as seen by the user. (Bottom Left) Overlay of the situational model on the console view. (Top Left) This is not an overlay, it is for the reader's benefit to show the user performing constrained teleoperation on the situational model.

the interaction forces. In our setup, we placed the force-torque sensor underneath the support plate holding the phantom, however, the system is capable of receiving force information over the network. The goal of the user is to locate the tumor using robotic exploration. With a conventional teleoperation system, the user would have to perform discrete probing on the phantom and do an offline estimation of the stiffness. With the help of CSA framework, the user simply moved in the lateral direction of the phantom maintaining contact with the surface at all times. The system superimposed a palpation motion based on sinusoidal force reference and recursively estimated/updated the stiffness map. This information was also displayed for the user to visualize. Further, if the user wants to narrow down the estimation of the stiffness to a specific region, the user provides a few point locations enclosing the region of interest (ROI). The system generated a Forbidden Region Virtual Fixture (FRVF), constraining the motion of the slave end effector inside the ROI. The compliance wrench was also calculated based on the interaction of the Proxy Slave with the situational model. The estimated compliance wrench was sent to the Master LLC to render force-feedback at the handle of the master console. Figure 11 shows the master console display as seen by the user. The situational model is located on the bottom left, which includes the rendering of the ROI (computed using convex hull) using points selected by the user.

Figure 12 shows the master console view of the teleoperation procedure. On the bottom left of the view is the virtual environment containing the situational model. The green dot represents the current tip position of the Proxy Slave. The red dot represents the closest point on the situational model from the tip position of the Proxy Slave. The red arrow on the situational model shows the current compliance force direction felt by the user on the master end-effector. The length of the arrow represents the magnitude of the force.

C. Extensions to dVRK

Here we provide a brief overview of a few features that were developed for CSA, some of which are already included in the current release of dVRK or will be available in future releases:

1) UDP Slave: The overall CSA concept is independent of the robotic platform. We currently support any type of slave device with the dVRK master robot for teleoperation. We have implemented a network-based slave component which act as a proxy for the real robot. The real system communicates with the proxy component based on User Datagram Protocol (UDP) using custom UDP packets. This allows the user to teleoperate a UDP-based slave robot with a dVRK master device. We chose UDP over existing middleware (*e.g.* ROS), to facilitate the integration with other robotic platforms (*e.g.* Matlab xPC). We have done some preliminary tests using this feature by teleoperating the IREP snake robot [33] with the dVRK master device at Vanderbilt University.

2) Simulated Mater/Slave: We have added support for the user to manipulate master or slave robots in simulation. This allows the users to test/debug various motion algorithms in simulation before testing on the real system.

3) Compliance wrench estimation: As mentioned earlier, we have incorporated support for compliance wrench estimation based on user specified gain parameters. This allows the user to formulate compliance type virtual fixtures on the master device.

IV. Discussion and Future Work

We believe our CSA framework is robust and will benefit in complex MIS procedures by providing proper guidance and surgical assistance. With our colleagues, we have demonstrated the significance of model-mediated teleoperation with virtual fixtures presented in this framework evaluated in a force controlled simulated ablation task [19]. We have incorporated continuous motion primitives which are time efficient palpation strategies compared to discrete probing. In our own work, we have developed a novel offline GP method to concurrently estimate the organ geometry and tissue stiffness based on continuous palpation [18]. In our current framework, we also have further improved our GP estimation (Section II-E) to provide real-time updates of the organ geometry and tissue stiffness while the surgeon is palpating the tissue. This online estimation allows the surgeon to have near to video frame rate updates of the stiffness map and make an informed decision on the trajectory of telemanipulation.

Our CSA framework is designed to enable other sensing modalities such as ultrasound images, confocal endomicroscopic images and other vision based information, however, the current implementation is limited to force sensing information. We also plan to evaluate the accuracy of our framework by incorporating it in other robotic platforms such as the IREP snake robot [33].

V. Summary

This paper has presented a software framework to provide situational awareness for the surgeon while teleoperating. Our software framework is implemented in a component based C++ framework. Support for ROS-based communication is provided to facilitate integration with other systems. We have demonstrated integration of CSA with the open source

dVRK hardware and software. Multiple useful features were developed for the dVRK community, which will be available in future releases. The software stack to provide CSA in a surgical assistant is maintained by JHU, with contributions from research groups at Vanderbilt University and Carnegie Mellon University.

VI. Acknowledgement

This work was supported in part by NSF NRI IIS1327566 and in part by Johns Hopkins University internal funds. The da Vinci Research Kit is supported by NSF NRI IIS1637789. We also thank our colleagues Nabil Simaan, Long Wang and Rashid Yasin at Vanderbilt University and Howie Choset, Elif Ayvali and Arun Srivatsan at Carnegie Mellon University, for their helpful insights while developing the ideas presented here.

References

[1] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable medical image registration: A survey," *IEEE Transactions on Medical Imaging*, vol. 32, no. 7, pp. 1153–1190, 2013.

[2] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, Dec 2010.

[3] S. Billings and R. Taylor, "Iterative most likely oriented point registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2014, pp. 178–185.

[4] I. Wanninayake, L. Seneviratne, and K. Althoefer, "Estimation of tissue stiffness using a prototype of air-float stiffness probe," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 1426–1431.

[5] D. Uribe, R. Stroop, T. Hemsel, and J. Wallaschek, "Development of a biomedical tissue differentiation system using piezoelectric actuators," in *Frequency Control Symposium, 2008 IEEE International*, May 2008, pp. 91–94.

[6] A. Sabatini, P. Dario, and M. Bergamasco, "Interpretation of mechanical properties of soft tissues from tactile measurements," in *Experimental Robotics I*, ser. Lecture Notes in Control and Information Sciences, V. Hayward and O. Khatib, Eds. Springer Berlin Heidelberg, 1990, vol. 139, pp. 452–462. [Online]. Available: http://dx.doi.org/10.1007/BFb0042534

[7] J. Dargahi, S. Najarian, V. Mirjalili, and B. Liu, "Modelling and testing of a sensor capable of determining the stiffness of biological tissues," *Electrical and Computer Engineering, Canadian Journal of*, vol. 32, no. 1, pp. 45–51, Winter 2007.

[8] H. Liu, D. P. Noonan, B. J. Challacombe, P. Dasgupta, L. D. Seneviratne, and K. Althoefer, "Rolling mechanical imaging for tissue abnormality localization during minimally invasive surgery," *Biomedical Engineering, IEEE Transactions on*, vol. 57, no. 2, pp. 404–414, 2010.

[9] M. Mahvash, J. Gwilliam, R. Agarwal, B. Vagvolgyi, L.-M. Su, D. D. Yuh, and A. Okamura, "Force-feedback surgical teleoperator: Controller design and palpation experiments," in *Haptic interfaces for virtual environment and teleoperator systems, 2008. haptics 2008. symposium on*, March 2008, pp. 465–471.

[10] G. Guthart and J. Salisbury, J., "The $INTUITIVE^{TM}$ telesurgery system: overview and application," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 618–621 vol.1.

[11] K. Xu and N. Simaan, "Intrinsic wrench estimation and its performance index for multisegment continuum robots," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 555–561, 2010.

[12] ——, "An investigation of the intrinsic force sensing capabilities of continuum robots," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 576–587, 2008.

[13] A. Bajo and N. Simaan, "Hybrid Motion/Force Control of Multi-Backbone Continuum Robots," *The International Journal of Robotics Research*, vol. 28, no. 9, pp. 1–13, July 2015.

[14] P. Mitra and G. Niemeyer, "Model-mediated telemanipulation," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 253–262, 2008.

[15] T. Xia, S. Leonard, I. Kandaswamy, A. Blank, L. Whitcomb, and P. Kazanzides, "Model-based telerobotic control with virtual fixtures for satellite servicing tasks," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[16] S. Sanan, S. Tully, A. Bajo, N. Simaan, and H. Choset, "Simultaneous compliance and registration estimation for robotic surgery," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[17] R. A. Srivatsan, E. Ayvali, L. Wang, R. Roy, N. Simaan, and H. Choset, "Complementary model update: A method for simultaneous registration and stiffness mapping in flexible environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 924–930.

[18] P. Chalasani, L. Wang, R. Roy, N. Simaan, R. H. Taylor, and M. Kobilarov, "Concurrent nonparametric estimation of organ geometry and tissue stiffness using continuous adaptive palpation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4164–4171.

[19] L. Wang, Z. Chen, P. Chalasani, R. M. Yasin, P. Kazanzides, R. H. Taylor, and N. Simaan, "Force-controlled exploration for updating virtual fixture geometry in model-mediated telemanipulation," *Journal of Mechanisms and Robotics*, vol. 9, no. 2, p. 021010, 2017.

[20] A. Deguet, R. Kumar, R. Taylor, and P. Kazanzides, "The cisst libraries for computer assisted intervention systems," in *MICCAI Workshop*, 2008.

[21] M. Y. Jung, M. Balicki, A. Deguet, R. H. Taylor, and P. Kazanzides, "Lessons learned from the development of component based medical robot systems," *J. of Software Engineering for Robotics (JOSER)*, vol. 5, no. 2, pp. 25–41, Sep 2014.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.

[23] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, February 1987.

[24] R. Featherstone, S. Sonck, and O. Khatib, *A general contact model for dynamically-decoupled force/motion control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 128–139. [Online]. Available: https://doi.org/10.1007/BFb0112956

[25] S. D. Billings, E. M. Boctor, and R. H. Taylor, "Iterative most-likely point registration (imlp): a robust algorithm for computing optimal shape alignment," *PloS one*, vol. 10, no. 3, p. e0117688, 2015.

[26] M. Kobilarov, "Cross-entropy motion planning," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[27] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization*. Springer, 2004.

[28] E. Ayvali, R. A. Srivatsan, L. Wang, R. Roy, N. Simaan, and H. Choset, "Using Bayesian optimization to guide probing of a flexible environment for simultaneous registration and stiffness mapping," *The International Conference on Robotics and Automation (ICRA)*, 2016.

[29] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," *CoRR*, vol. abs/1707.04294, 2017. [Online]. Available: http://arxiv.org/abs/1707.04294

[30] E. Ayvali, A. Ansari, L. Wang, N. Simaan, and H. Choset, "Utility-guided palpation for locating tissue abnormalities," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 864–871, April 2017.

[31] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the $daVinci^{TM}$ surgical system," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6434–6439.

[32] Z. Chen, A. Deguet, R. H. Taylor, and P. Kazanzides, "Software architecture of the da Vinci Research Kit," in *Robotic Computing (IRC), IEEE International Conference on*. IEEE, 2017, pp. 180–187.

[33] A. Bajo, R. E. Goldman, L. Wang, D. Fowler, and N. Simaan, "Integration and preliminary evaluation of an insertable robotic effectors platform for single port access surgery," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3381–3387.