Modeling of Surgical Procedures Using Statecharts for Semi-Autonomous Robotic Surgery

Fabio Falezza, Nicola Piccinelli, *Member, IEEE*, Giacomo De Rossi, Andrea Roberti, Gernot Kronreif, Francesco Setti, Paolo Fiorini, *Life Fellow, IEEE*, and Riccardo Muradore, *Member, IEEE*

***Abstract*—In this paper we propose a new methodology to model surgical procedures that is specifically tailored to semi-autonomous robotic surgery. We propose to use a restricted version of statecharts to merge the bottom-up approach, based on data-driven techniques (e.g., machine learning), with the top-down approach based on knowledge representation techniques. We consider medical knowledge about the procedure and sensing of the environment in two concurrent regions of the statecharts to facilitate re-usability and adaptability of the modules. Our approach allows producing a well defined procedural model exploiting the hierarchy capability of the statecharts, while machine learning modules act as soft sensors to trigger state transitions. Integrating data driven and prior knowledge techniques provides a robust, modular, flexible and re-configurable methodology to define a surgical procedure which is comprehensible by both humans and machines. We validate our approach on the three surgical phases of a Robot-Assisted Radical Prostatectomy (RARP) that directly involve the assistant surgeon: bladder mobilization, bladder neck transection, and vesicourethral anastomosis, all performed on synthetic manikins.**

***Index Terms*—Surgical robotics, statecharts, supervisory controller, autonomous robotics.**

I. Introduction

THE RESEARCH interest in Robotic-assisted Minimally Invasive Surgery (R-MIS) is shifting from teleoperated devices to the development of autonomous support systems for the execution of repetitive surgical steps, such as suturing, ablation and microscopic image scanning. The higher level of autonomy can potentially further improve the quality of an intervention in terms of patient's safety and recovery time [1]. Moreover, it can optimize the use of operating rooms, reducing the surgeon's workload and therefore hospital costs. In general, autonomy requires systems with advanced capabilities in perception, reasoning, decision making [2], motion planning [3] and interaction with the physical environment. Nonetheless, for autonomous or semi-autonomous systems Human Robot Interaction (HRI) plays a key role in providing both safety of execution and a successful knowledge transfer between users and robots. Two different approaches can be adopted to model the medical knowledge from the surgeons: a top-down and a bottom-up approach.

The *top-down approach* is based on encoding prior knowledge into a formal representation understandable by both humans and machines. Different approaches have been proposed, like description logic [4], formal ontologies [5], or defeasible reasoning [6]. Statecharts models are a graphical specification formalism that allows the nesting of Finite State Machines (FSMs hierarchy), their orthogonality (FSMs parallelism) and re-usability of components [7], [8]. The major advantage brought by FSMs is that they can be formally verified [9], and, therefore, are always guaranteed to operate according to their design. For this reason, FSMs are widely employed in the representation of mission-critical workflows, such as the case for surgical procedures. The representation power of statecharts has been exploited to build a discrete-event simulation model of the pre-operative process [10]. The progress of individual patients through surgical care is decribed as series of asynchronous updates in patients' records; these updates are triggered by events produced by parallel FSMs that represent concurrent clinical and managerial activities. The *bottom-up approach* tries to infer a model from raw data through data analysis techniques, such as deep learning, possibly in an unsupervised, end-to-end manner to speed-up the process and to avoid labeling bias [11]. In this work, we adopt a safer approach that follows the *engineering stack* guidelines for which the top down model is adapted in its formulation to the events based on the available observations on both the environment and the robots [12]. This improves both safety and explainability of possible Machine Learning (ML) modules used to process the data. Relevant information can be extracted from documents [13] or from video streams [14], also combined with instrument usage signals (e.g., kinematics in case of robotic surgery) [15]. The work in [16] adopts multiple disjointed FSMs, one for each surgical subtask, where the parameters (i.e., the thresholds viable to trigger events) are learned in a reinforcement learning

Manuscript received May 7, 2021; revised July 12, 2021; accepted August 23, 2021. Date of publication September 6, 2021; date of current version November 19, 2021. This article was recommended for publication by Associate Editor A. Casals and Editor P. Dario upon evaluation of the reviewers' comments. This work was supported in part by the European Research Council (ERC) through the European Unions Horizon 2020 Research and Innovation Programme under Grant 742671 (ARS), and in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 779813 (SARAS). *(Corresponding author: Fabio Falezza.)*

Fabio Falezza, Nicola Piccinelli, Giacomo De Rossi, Andrea Roberti, Francesco Setti, Paolo Fiorini, and Riccardo Muradore are with the Department of Computer Science, University of Verona, 37134 Verona, Italy (e-mail: fabio.falezza@univr.it; nicola.piccinelli@univr.it; giacomo.derossi@univr.it; andrea.roberti@univr.it; francesco.setti@univr.it; riccardo.muradore@univr.it).

Gernot Kronreif is with ACMIT Gmbh, 2700 Wiener Neustadt, Austria (e-mail: gernot.kronreif@acmit.at).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMRB.2021.3110676, provided by the authors.

Digital Object Identifier 10.1109/TMRB.2021.3110676

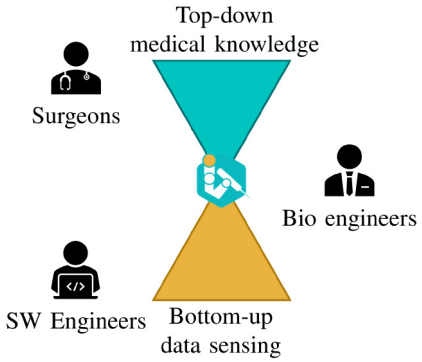


Fig. 1. The proposed methodology from a perspective of knowledge integration (top-down and bottom-up approaches) and required technical skills. This approach is the one followed in the EU funded SARAS project (`www.saras-project.eu`).

manner. In this paper we propose a methodology based on a *revised statechart model* that aims at finding a proper merging of top-down and bottom-up approaches to implement a control strategy for semi-autonomous surgical operations, as depicted in Fig. 1. The statechart formulation has been selected for its inherent combination of structured and unstructured knowledge that can be handled hierarchically at different levels of abstraction. To achieve this abstraction, the statechart at the top level is split in two concurrent regions: procedure and observer. The *procedure region* represents the medical knowledge extracted from clinical trials with surgeons and from literature review; the *observer region* is composed of a concurrent set of FSMs that provides a logical description for the environment state (e.g., semantic robot position, kinematics state, etc). The ML-based bottom-up approach is considered as a software-sensor that operates uniquely within each separate observer of the region to provide trigger events. The decision on how such triggers drive the procedure evolution over time is controlled by the structure of the procedure region. The adoption of data-driven modules to provide feedback regarding the environment allows increasing flexibility and modules' re-usability for the entire system. The procedure region of the statechart is subdivided in hierarchical levels to refine how the desired behaviour of the robotic system is defined. Such division of knowledge at the procedure design level helps in simplifying both modelling and comprehension of the statechart itself. The control commands for the robots are then defined in the innermost level. In summary, as main contributions in this paper we present:

- a robust, modular, flexible and re-configurable methodology to define a surgical procedure,
- an effective approach to integrate top-down surgical knowledge with bottom-up sensor data,
- a supervisory technique to control semi-autonomous robotics systems.

The paper is organized as follows: in Section II we will review the state of the art in task automation. In Section III we will detail the proposed modelling methodology and in Section IV we will present the case study. In Section V the experimental setup, shown in Fig. 6, used to validate our architecture is described and the execution of a specific phase is presented. Finally, in Section VI we will discuss the overall tests conducted with surgeons and few conclusions are drawn on the proposed approach and future perspectives are outlined.

II. Related Work

The automation of basic surgical tasks is taking a lot of attention in the recent years with various approaches being proposed. For instance, the description using Finite State Machine (FSM) can be used to account for simple tasks, where the environment is assumed to be static and situation awareness can be neglected. In [17] an automated mechanical needle guide to improve precision is proposed. In [18] an FSM based framework for automation of surgical sub-tasks is developed and tested on simple surgeon training tasks like peg&ring and knot-tying. In [19] a depth-sensor has been used to increase the accuracy of peg&ring task. Finally, in [20], a Hierarchical Finite State Machine (HFSM) is exploited to control autonomous mobile systems. This work shows how the hierarchy can be exploited to subdivide the controller between discrete and continuous time, allowing the separation of the high-level decision-making to their low-level implementation.

An alternative approach to FSM is represented by Behaviour Trees (BT) with their emphasis on modularity [21]. In [22] a BT is used to model and control a semi-autonomous simulated brain tumor ablation; here the leaves of the BT represent the surgical sub-tasks of the ablation. The main drawback of BTs is the lack of introspection. Indeed, the current state of the system cannot be retrieved directly, but it must be derived analysing the path from the root node to the current running leaf. In [23] a cognitive framework to perform autonomous needle insertion has been proposed. The supervisory controller was built by using a Hidden Markov Model.

All of these data-driven approaches, however, require a large amount of data to achieve a sufficiently robust learning, which is not usually available in surgery. Moreover, the planned action cannot be easily interpreted and monitored by a human expert since data-driven models are often based on latent variables representation of the environment.

In critical scenarios like surgical procedures, knowledge based approaches are the preferred way since they provide a clearer description of the workflow. For instance, in [24] an ontology-based framework for the automation of the peg&ring task has been proposed. The main drawback was the lack of real-time reconfiguration of the system. In fact, ontologies are much more used in the field of situation understanding by humans [25]. A solution to the limitation of the ontologies can be found in the non-monotonic programming, where the planning is carried out in a more flexible way, thus the knowledge can be updated in real-time from the sensing information. In [2] Answer Set Programming has been used to define the reasoning module and has been successfully applied on an automated peg-and-ring task. The drawback of non-monotonic programming resides in the computational complexity required to solve a planning problem, which makes this approach often unsuitable for real-time applications.

III. Method

The proposed modelling methodology relies on the statecharts visual notation, a simplified example of which is

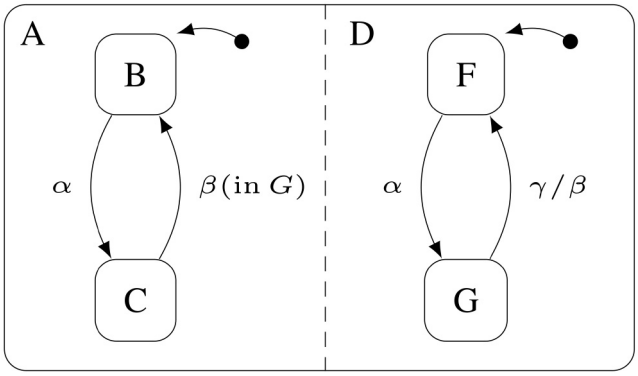


Fig. 2. An example of a two-level and concurrent statechart.

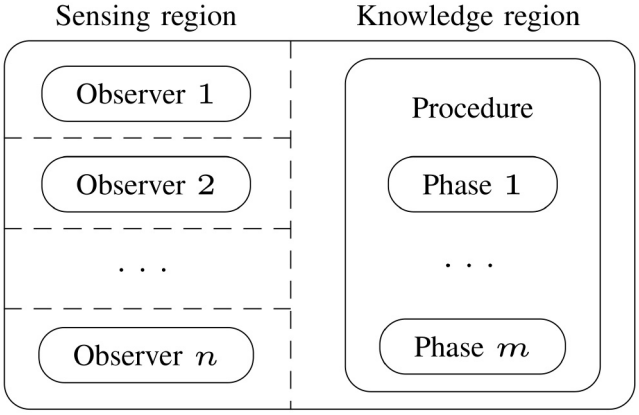


Fig. 3. The structure of the revised statechart. The statechart is composed of n observers that run concurrently to the procedure. The observers generate events consumed by the procedure's state transitions. The procedure hierarchy is not reported in this figure but is composed of three levels.

shown in Fig. 2, that will be briefly summarized to ease the reading in the following sections. A statechart uses rounded rectangles to denote the states (or regions) at any level, using encapsulation to express the hierarchy relation. Arrows are allowed to start and terminate at any level. Each arrow is labelled with an event, optionally with a parenthesized condition and an output event after a backslash, and marks the possible transitions between states. The initial state is marked with an arrow pointing to it starting from a small dot without label. The encapsulations imply the exclusive-or (XOR) decomposition of inner states, meaning that the current state must be only one and it must be selected from the states set of the grouping state. Moreover, the encapsulations could introduce the AND decomposition, capturing the property that, being in a state, the system must be in all of its AND components using dashed lines.

The statechart notation is quite expressive as it allows modeling behaviours that are not required in our application. Therefore, in this paper we propose a revision of statecharts which impose a clear separation between the sensing of the environment and the representation of the procedure knowledge into two distinct *regions* of the chart, as shown in Fig. 3. Indeed, the *concurrency* capability of the statecharts will be exploited only to represent the parallelism of the sensing, while the *hierarchy* will be used only to separate the medical knowledge from the implementation of the robotic tasks. The proposed revisions are:

- R_1 The hierarchy depth of the procedure must be equal to three: *activity*, *task* and *primitive* (see Section III-A);
- R_2 At the lowest level (i.e., primitive), the states are atomic and hierarchy-agnostic, thus they do not depend on the execution time or the order with respect to others states (see Section III-B);
- R_3 The sensing region is evaluated concurrently to the knowledge region, but it is the only generator of events during transitions (see Section III-C);
- R_4 Concurrency is only allowed within the sensing region, thus transitions between concurrent FSMs are forbidden (see Section III-D).

Using our revised statechart modelling, we can now define the adopted heuristics for the planning. We exploit the intrinsic priority of the hierarchy since an event generated by the sensing system can be adopted as a transition trigger by any level of the procedure. At each control cycle, the controller, which evaluates the statechart, starts from higher to lower level and checks if there exists a trigger event on the edges exiting the current state. If a trigger event is present, then the state is activated and the search is stopped. When a transition occurs, the innermost FSMs are reset to their initial state. Even if such heuristic ensures the high-level command to be prioritized, it can lead to transitions shadowing, meaning that transitions on lower-levels could never been crossed if transitions at higher-levels on the same trigger are fired. To minimize such effect, the modelling of the procedure should be approached exploiting a semantic separation of events. At the higher levels of the procedure, the events should refer only to the environment conditions for the activity (e.g., in surgery, positioning over the bladder, thread cut, catheter visible, etc), while at the lowest level the events should refer only to the robot internal state (e.g., target pose reached, pose not reachable, grasp closed, etc). Fig. 3 shows an example of the structure of our revised statechart.

The extension to multiple robots can be obtained by simply duplicating the statechart and adjusting it for each manipulator. The synchronous operation of multiple statecharts that operate on the same triggers is guaranteed by the assumption that each trigger produced by every observer is processed simultaneously by all statecharts. Therefore, each robot effectively observes the environment, including all others robots, independently, thus eliminating the necessity of specific synchronization states.

A. Procedure Depth

Generally, the number of hierarchical levels allowed in a statechart is unbounded. In our case, as stated in the rule R_1 the number of the levels is set to three. The adoption of a three-level hierarchy has been proposed in [26] and is also part of the definition of the Hierarchical Task Network (HTN) [27], which is a popular task planning methodology. Therefore, the resulting procedure can be defined by grouping the states of the statechart into three well-defined levels

of a surgical procedure: **phase**, **action** and **surgeme**, mapped respectively into *activity*, *task*, and *primitive*, which is a lexical formalism also adopted in [28].

The highest level, composed of phase states, is the equivalent to the goals of the STRIPS modelling [29]. In the surgical field, this level models the main phases of a complex surgical procedure (e.g., in the case of radical prostatectomy, some goal tasks could be bladder mobilization, bladder neck transection, or an idle statement). The middle level, composed of action states, can be seen as a set of intermediate tasks that make the goal task (e.g., for the bladder neck transection phase a set of actions are grasping the catheter and pulling the catheter).

Finally, in the lowest level we find atomic sub-task that cannot be logically subdivided into a smaller subset, thus composed of surgeme states (e.g., close the gripper or move to a specific point).

B. Atomic and Hierarchy-Agnostic Execution

The rule R_2 enforces the re-usability of the surgemes allowing the definition of multiple complex surgical procedures by means of a relatively small set of common and shared primitives. The side effect of this design choice is the need of prohibiting any transition from a lower level to a higher one. In fact, the re-usability requires to have hierarchy-agnostic surgemes and of course transitions towards the upper levels requires a knowledge about the parent states (not available at the surgeme level).

C. Event Generation

An observer is a FSM that operates concurrently to the procedure with the aim of observing the environment and of generating trigger events based on measurements (observations). The definition of the observer entities allows separating the generation of triggers, which happens in the sensing region, from their consumption, which happens in the knowledge region. Therefore, the rule R_3 states that the knowledge region must not generate triggers to easily ensure the absence of infinite loops, undefined behaviors, and deadlocks. The definition of the events can be obtained in two different ways:

- *bottom-up:* the procedure is modeled and refined by surgeons using only the available observers;
- *top-down:* the procedure is first defined based on the knowledge of surgeons, then the engineers will develop the requested observers to accomplish the procedure.

The formalism presented hitherto intends to logically separate these two approaches to avoid contrasts in requirements formulated by surgeons and engineers: the surgeons designing the procedure in a top-down manner could specify undetectable events (or that could require hardware that is not applicable to laparoscopy), whereas the engineers working in a bottom-up manner could overlook important events that do not rise directly from the data. This approach simplifies the modelling of the procedure and allows to design a more efficient system based on the available sensors, surgical instruments, and computational power. The managing of the interleaved requirements should be assigned to a specialized bio-engineer who has to handle surgical and engineering critical aspects.

D. Concurrent Isolation

The standard set of rules for statecharts do not prevent the definition of edges over concurrent states, but in the case of the proposed procedure-observers statecharts this kind of transitions are meaningless as surgical procedures follow a well-defined sequence of states (which is hampered only by the occurrence of unexpected complications that must be handled manually). This is the reason why we introduced rule R_4 that forbids the definition of edges on concurrent states, thus between observers and procedures, and between different observers.

IV. Case Study: Radical Prostatectomy

Robotic Assisted Radical Prostatectomy (RARP) is a surgical procedure where the surgeon utilises a robotic manipulator to remove the prostate along with, in some cases, the seminal vesicles and the pelvic lymph nodes [30]. The procedure is performed for treatment of prostate cancer. All results of robotic prostatectomy so far indicate the benefits of minimally invasive surgery while also showing encouraging short and long-term outcomes in terms of continence, potency, and cancer control; it is regarded as a major innovation in the surgical treatment of prostate cancer. RARPs are currently performed using either the da Vinci surgical system or any comparable robotic platform. Surgeons remotely control the instruments of the robotic manipulator using two joysticks available on the console. In the operating room, there must be also an assistant surgeon next to the patient, helping the main surgeon. This paper provides a first step towards the full automation of the assistant surgeon's role.

We followed the procedure presented in [31], [32] that bridges the clinical and engineering requirements to improve the effectiveness of both the surgeon, equipped with two da Vinci instruments and an endoscope, and the semi-autonomous assistant handling two standard laparoscopy instruments. At first, the surgeon identifies the proper plane of dissection to operate on the bladder neck, which is then divided transversely with respect to the urethra, until he/she identifies the urethral catheter pushed through the prostate. At this point, the assistant surgeon, using the right laparoscopic tool, mobilizes the bladder to clear the view, and, with the left laparoscopic tool, raises the prostate. Once the prostate is suspended anteriorly, the main surgeon grasps the tip of the catheter and lifts it upwards to increase access to the lower part of the prostate, including the vas deferens and the seminal vesicles. After the prostate has been removed, the main surgeon performs the vesicourethral anastomosis. During this phase, the activities of the assistant surgeon consist of avoiding the bladder inflation by keeping it pushed down and, once the suture has been completed, cutting the needle's thread with the scissors [30]. We have distributed the activities of the assistant between the two robotic arms: the left side arm is in charge of the bladder neck transection phase, provided with a grasper tool for mobilizing the catheter, while the right side arm takes

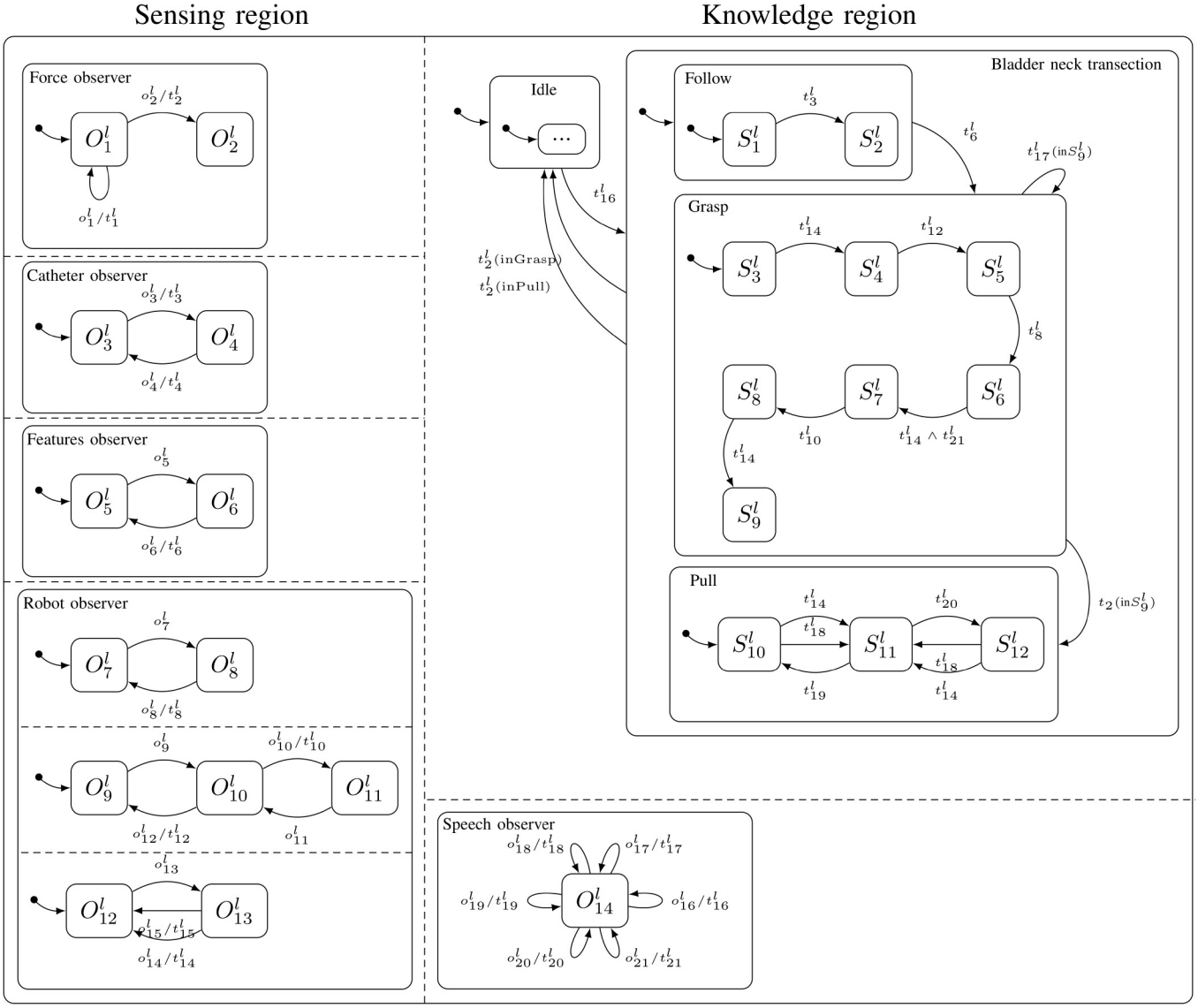


Fig. 4. Left arm procedure modeled as statechart (i.e., bladder neck transection). The observer states are represented by O_i, the surgeme states are represented by S_j and the observed events and the triggers are labeled with o_k and t_k respectively.

care of the bladder mobilization and anastomosis with a pair of curved scissors, which are curved upward during the former phase to avoid puncturing the bladder; all motions are intended to be executed with collision-free trajectories considering both the main surgeon's instruments and the anatomy as obstacles. The anastomosis for the assistant consists only in cutting the thread, which translates into movements toward the thread and then closing the instrument to cut.

In the following paragraphs, we will describe the statecharts used to control both the manipulators. These charts represent a single instance of operation modeled on the experimental setup: they demonstrate the general methodology for merging the top-down medical knowledge with the bottom-up sensing data. Initially we will focus on the sensing part describing in detail which observers have been integrated and which events have been observed. Subsequently, the surgemes are defined and integrated into the platform. Finally, the procedures for the left and right arm are modeled following the proposed methodology. The procedures schemes are shown in Fig. 4 and Fig. 5, with each symbol explained in Table I.

A. Observers

Each *robot observer* is a statechart composed of 3 concurrent FSMs. The purpose of these observers is to identify the current state of the robot and to trigger events when a desired state is reached. The observed states are the Cartesian position of the manipulator's end-effector, the rotation of the tool and the closure/opening state of the gripper. For instance, when the target rotation of the left arm is reached, the trigger t^l_8 is generated; when the tool is opened or closed, the triggers t^l_{12} or t^l_{10} are generated respectively; t^l_{14} or t^l_{15} are generated when the target pose is reachable or not, respectively. The triggers for the right arm work in the same manner, please see Table I for the full list.

The *force observer* is made of an FSM with only two states. The purpose of this observer is to measure the interaction force

TABLE I
SYMBOLS DESCRIPTION OF THE OBSERVERS, SURGEMES AND TRIGGERS FOR THE LEFT (l, TOP) AND RIGHT (r, BOTTOM) ARMS IN FIG. 4 AND FIG. 5

Observer	Observer Label	Surgeme	Surgeme Label	Event/Trigger	Event/Trigger Label
O_1^l	Force under threshold	S_1^l	Idle	o_1^l/t_1^l	Force under threshold
O_2^l	Force over threshold	S_2^l	Follow the catheter	o_2^l/t_2^l	Force over threshold
O_3^l	Looking for catheter	S_3^l	Approach the catheter	o_3^l/t_3^l	Catheter found
O_4^l	Tracking the catheter	S_4^l	Open the tool	o_4^l/t_4^l	Catheter vision lost
O_5^l	Features under threshold	S_5^l	Rotate the tool	o_5^l	Velocity over threshold
O_6^l	Features over threshold	S_6^l	Move to the grasping point	o_6^l/t_6^l	Catheter stopped
O_7^l	Rotation in idle	S_7^l	Close the tool	o_7^l	Tool is rotating
O_8^l	Rotation moving	S_8^l	Check the grasping	o_8^l/t_8^l	Goal rotation reached
O_9^l	Tool close	S_9^l	Wait for confirm grasping	o_9^l	Tool is closing
O_{10}^l	Tool moving	S_{10}^l	Pull more	o_{10}^l/t_{10}^l	Tool closed
O_{11}^l	Tool open	S_{11}^l	Idle	o_{11}^l	Tool is opening
O_{12}^l	Robot idle	S_{12}^l	Pull less	o_{12}^l/t_{12}^l	Tool open
O_{13}^l	Robot moving			o_{13}^l	Robot is moving
O_{14}^l	Speech recognition			o_{14}^l/t_{14}^l	Goal reached
				o_{15}^l/t_{15}^l	Goal not reachable
				o_{16}^l/t_{16}^l	Speech grasp catheter
				o_{17}^l/t_{17}^l	Speech retry
				o_{18}^l/t_{18}^l	Speech stop
				o_{19}^l/t_{19}^l	Speech pull more
				o_{20}^l/t_{20}^l	Speech pull less
				o_{21}^l/t_{21}^l	Speech close instrument
O_1^r	Preoperative poses	S_1^r	Approach bladder	o_1^r/t_1^r	On bladder top
O_2^r	Rotation in idle	S_2^r	Idle	o_2^r/t_2^r	On bladder bottom
O_3^r	Rotation moving	S_3^r	Push bladder	o_3^r/t_3^r	On safe pose
O_4^r	Tool close	S_4^r	Hold bladder	o_4^r	Tool is rotating
O_5^r	Tool moving	S_5^r	Push less	o_5^r/t_5^r	Goal rotation reached
O_6^r	Tool open	S_6^r	Push bladder here	o_6^r	Tool is closing
O_7^r	Robot idle	S_7^r	Release bladder	o_7^r/t_7^r	Tool closed
O_8^r	Robot moving	S_8^r	Safe pose	o_8^r	Tool is opening
O_9^r	Speech recognition	S_9^r	Follow thread	o_9^r/t_9^r	Tool open
		S_{10}^r	Open scissor	o_{10}^r	Robot is moving
		S_{11}^r	Rotate scissor	o_{11}^r/t_{11}^r	Goal reached
		S_{12}^r	Approach thread	o_{12}^r/t_{12}^r	Goal not reached
		S_{13}^r	Cut thread	o_{13}^r/t_{13}^r	Speech bladder push down
				o_{14}^r/t_{14}^r	Speech cut needle
				o_{15}^r/t_{15}^r	Speech idle
				o_{16}^r/t_{16}^r	Speech close
				o_{17}^r/t_{17}^r	Speech restart
				o_{18}^r/t_{18}^r	Speech push more
				o_{19}^r/t_{19}^r	Speech push less
				o_{20}^r/t_{20}^r	Speech push here
				o_{21}^r/t_{21}^r	Speech release
				o_{22}^r/t_{22}^r	Speech stop

of the autonomous robotic arm to verify whether a successful grasping happened. A parametric threshold W_t is set: if the measured force $W_m > W_t$ the t_2^l trigger is generated while transiting from state O_1^l to O_2^l, otherwise it loops on the initial state O_1^l generating the trigger t_1^l continuously.

The *catheter observer* and the *feature observer* are two concurrent FSMs composed of two states with transitions that allow to loop in both states. The former analyses the camera RGB images to detect the catheter: when the catheter is found the transition from O_3^l to O_4^l is performed, triggering the event t_3^l; when the track is lost the opposite state transition is computed triggering the event t_4^l. More details on how the catheter detection is computed are presented in the Supplementary Material. The feature observer estimates the catheter odometry and checks whether the linear speed of the catheter goes over and then below two parametric thresholds W_t^{max} and W_t^{min} within a sliding window T_t. This observer is used during the procedure to detect whenever the main surgeon has completely extracted the catheter from the prostate and is ready to give it to the autonomous robotic arm. Practically, if the measured speed $W_m > W_t^{max}$ and then within T_t seconds $W_m < W_t^{min}$, the transitions from O_5^l to O_6^l and then back to O_5^l are performed, triggering the event t_6^l.

An instance of the *speech observer* is available to each arm and they are FSMs composed of a single state (e.g., O_{14}^l, O_9^r). The state runs a speech recognition algorithm that adopts Natural Language Processing (NLP) to transform commands imparted by the surgeon's speech to triggers.

The *preoperative observer* is in charge to generate a trigger when the robot's end-effector reaches a predefined position in the anatomy. The FSM is composed of a single state O_1^r, with as many transitions as the number of predefined positions. For instance, for the right arm there are three transitions t_1^r, t_2^r

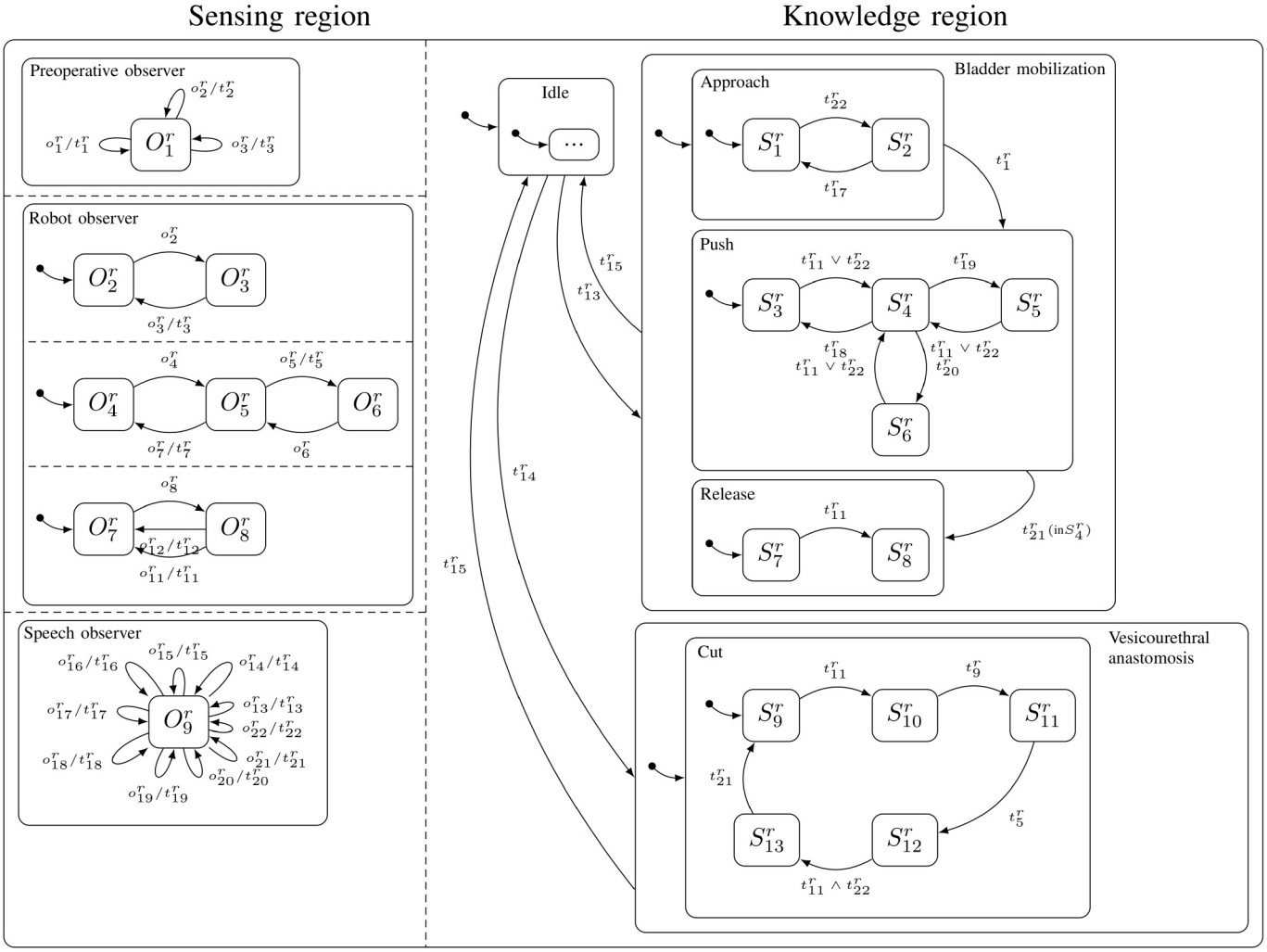


Fig. 5. Right arm procedure modeled as statechart (i.e., bladder mobilization, vesicourethral anastomosis). The observer states are represented by O_i, the surgeme states are represented by S_j and the observed events and the triggers are labeled with o_k and t_k respectively.

describing landmarks on the bladder and t_3^r which describes the idle position for the arm.

B. Surgemes

Only 4 types of surgemes have been implemented to accomplish the whole procedure under study, and they are the following:

- *No operation surgeme* is used to keep the robot in the current state. No operation is performed until a transition is done. This surgeme is used in the states S_1^l, S_9^l, S_{11}^l, S_2^r, and S_4^r.
- *Movement surgeme* interpolates a trajectory from the current position to the target position. The target position is given as a required parameter to the surgeme. If the target position is changed at run-time the trajectory is recomputed. This surgeme is used in the states S_2^l, S_3^l, S_6^l, S_8^l, S_{10}^l, S_{12}^l, S_1^r, S_3^r, S_5^r, S_6^r, S_7^r, S_8^r, S_9^r, and S_{12}^r.
- *Rotate tool surgeme* is used to rotate the tool around its main axis. It takes as argument the desired absolute rotation. This surgeme is used in the states S_4^l and S_{11}^r.
- *Open/close surgeme* is used to open or close the grasper/scissors. It takes as argument the desired target closing percentage, e.g., 100% means totally closed. This surgeme is used in the states S_5^l, S_7^l, S_{10}^r, and S_{13}^r.

C. Left Arm Procedure

The procedure for the left arm is composed of two phases: the *idle phase* and the *bladder neck transection phase*. Since the experiment is performed in a semi-autonomous scenario, while the main surgeon is dissecting the bladder neck, the assistant surgeon does not have to accomplish any specific task. For this reason, in the *idle phase* the tool stays in a safe position allowing the main surgeon to operate. In this phase, the autonomous system is waiting for the trigger t_{16}^l which corresponds to the speech command left arm grasp the catheter, which allows the phase transition to the *bladder neck transection phase*.

The *bladder neck transection phase* is composed of three actions: *follow action*, *grasp action* and *pull action*. The first action has the purpose to follow the catheter by keeping a safe distance, while the second one is designed to grasp the catheter. The last action pulls the catheter towards the pelvis bones and handles any possible positioning adjustments requested by the main surgeon. The transition from *follow*

action to *grasp action* is enabled when the catheter stops after reaching a threshold velocity (trigger t_6^l). The *grasp action* progressed only into three possible transitions which are dependent on the internal current state S_9^l: if t_{17}^l is triggered by the surgeon command "left arm retry", the effect is a reset of the internal state to the initial one of the action; if the measured force on the tool is under threshold, t_1^l is triggered and the state doesn't change as it waits for a slight pull by the surgeon to verify the grasp; finally, if the force is over threshold (t_s^l), the current action is changed to *pull* as the grasp is considered as achieved.

Follow Action: This is composed of two surgeme S_1^l and S_2^l. The system remains in the initial state S_1^l until the catheter becomes visible (trigger t_3^l), a condition that fires the state transition to S_2^l in which the left SARAS (SARAS2) arm follows the catheter within a predefined safe distance from the surgeon's tool currently grasping the catheter.

Grasp Action: This is composed in total of seven surgemes. The initial one (S_3^l) consists of an additional movement towards the catheter to signal the surgeon that the correct sequence of events towards the grasping action is occurring. When the tool reaches the desired position (trigger t_{14}^l), the next state is set to S_4^l to rotate the tool relative to the catheter's attitude. Once the desired rotation has been reached (t_8^l), the controller transits to the state S_5^l where the manipulator opens the grasper. At this point, when the grasper has been successfully opened (t_{12}^l), the next surgeme in S_6^l moves the tool to the final grasping point, which is computed as the current position of the da Vinci tooltip, with a small offset to avoid tool collisions. Now, the system can transit to S_7^l whenever either the desired grasping point is reached (t_{14}^l) or the speech command "left arm close the instrument" (t_{21}^l) is requested. In this surgeme (S_7^l) the left arm closes the grasper and, once the tool has been totally closed (t_{10}^l) the next state is set to S_8^l. At this point, the tool is commanded back slightly along the RCM axis to carefully pull the catheter and thus verify whether the grasping was truly successful. When the desired position is reached (t_{14}^l) the system transits to the next state S_9^l which is a *no operations surgeme*.

Pull Action: This is composed of three surgemes, with the initial state S_{10}^l used to move the manipulator towards a point defined as the maximum traction point (from the pre-registered map). The transition to the next state S_{11}^l could happen if the desired position is reached (trigger t_{14}^l) or the "left arm stop" speech command (t_{18}^l) is provided by the main surgeon. The state S_{11}^l is a *no operations surgeme* and it is intended to be the final surgeme of the procedure. If the speech command "left arm pull more" (t_{19}^l) is provided, the next state goes back to S_{10}^l; if the speech command is "left arm pull less" (t_{20}^l), the next state is set to S_{12}^l. This state moves the tool to a point nearby the bladder neck to release the catheter, and the next state is set to S_{11}^l if the desired position is reached or if the "left arm stop" sentence t_{18}^l is detected.

D. Right Arm Procedure

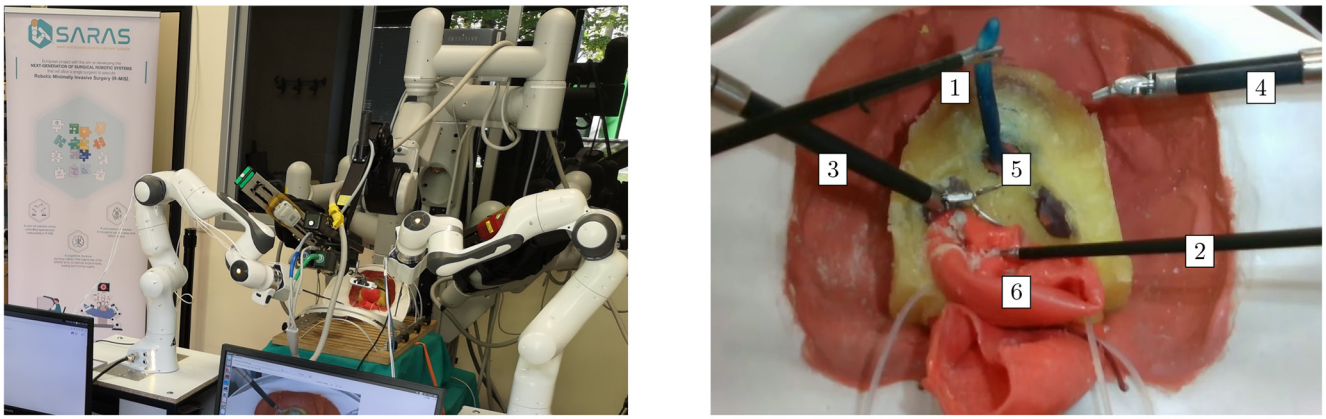
The procedure for the right arm is limited to three phases due to the scissors tool: these are the *idle*, the *bladder mobilization*, and the *vesicourethral anastomosis* phases. The *idle phase* is again used to keep the robot in a safe configuration and far from the main surgeon's tools. When the "right arm push down the bladder" speech command is provided (trigger t_{13}^r) the system transits to the *bladder mobilization phase*. This phase is composed of three actions: *approach action*, *push action* and *release action*. The first action approaches the upper part of the bladder, while the second action pushes the bladder down and handles any possible adjustment of the applied pressure or pressure point. The last action releases the bladder and moves back the manipulator to the safe position. The transition from the *approach action* and the *push action* happens when the end-effector reaches the top of the bladder (t_1^r). The transition from *push action* to *release action* occurs when the "right arm release" command is provided (t_{21}^r). When trigger t_{14}^r is fired, following the "right arm cut the needle" speech command, a phase change occurs towards the *vesicourethral anastomosis phase*. This phase, the last one of the procedure, is composed of a single action *cut action* consisting of moving the scissors to the thread and cut it.

Approach Action: This is composed of two surgemes S_1^r and S_2^r. The initial state is S_1^r where the end-effector is moved to the top of the bladder. In case the surgeon needs to stop the motion, the speech command "stop" (t_{22}^r) changes the current surgeme to S_2^r. In the state S_2^r the robot is still and it keeps this state until the speech command "right arm retry" is provided (t_{22}^r).

Push Action: This is composed of four surgemes, where the initial state S_3^r moves the robot to the bottom of the bladder increasing the pressure applied until either a "stop" speech command is received (t_{22}^r) or the bottom position is reached (t_{11}^r). In any case, the next state S_4^r keeps the robot still and holds the bladder down. From this state three speech commands could fire a state transition: "right arm push more" (t_{18}^r) which sets the next state back to S_3^r; "right arm push less" (t_{19}^r) which let to transit to the state S_5^r or "right arm push here" (t_{20}^r) which trigger the transition to the state S_6^r. The state S_5^r moves the arm to the top of the bladder, decreasing the applied pressure until either a "stop" speech command is received (t_{22}^r) or the position is reached (t_{11}^r). The state S_6^r moves the arm towards the main surgeon tool and is used to apply pressure to a different point. When the "stop" speech command is received (t_{22}^r) or the goal is reached (t_{11}^r) it transits back to S_4^r.

Release Action: This is composed of two surgemes S_7^r and S_8^r. The initial state S_7^r moves the tool back on the top of the bladder. When the desired position is reached (t_{11}^r), the state transits to S_8^r which moves the tool to the safe position.

Cut Action: This is composed of five surgemes. The initial state S_9^r moves the robotic arm towards the position of the thread (which is in our case pointed by the main surgeon's tool). When the desired thread position is reached (t_{11}^r), the system transits to S_{10}^r where the scissors are opened. When the tool is fully open, the trigger t_9^r is generated and the next state is set to S_{10}^r where the tool is rotated to be perpendicular to the thread. Once the desired rotation is reached (t_5^r), the state is set to S_{12}^r and the manipulator approaches the thread. In case



(a) External view (b) Anatomical phantom

Fig. 6. The platform used for the experimental validation: (1) the autonomous left robotic arm (SARAS2) pulling the catheter, (2) the autonomous right robotic arm (SARAS1) mobilizing the bladder, (3) the left teleoperated da Vinci arm (PSM2), (4) the right teleoperated arm da Vinci (PSM1), (5) the bladder neck, (i.e., the conjunction between bladder and prostate), (6) the mobilized bladder.

the thread position is reached (t^r_{11}) or the speech command "right arm close the instrument" (t^r_{22}) is requested by the main surgeon, the next state is set to S^r_{13} and the robot cuts the thread. If the thread is missed, the surgeon can request by the command "right arm release" (t^r_{21}) to restart the action from S^r_9, otherwise the phase is completed.

V. Experimental Validation

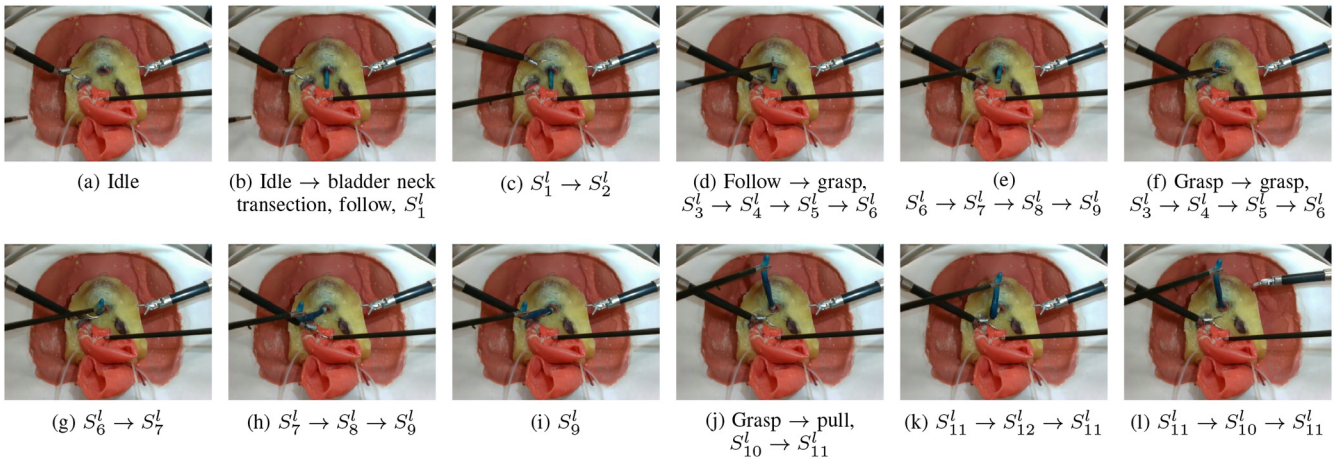
The proposed modeling approach has been tested on a realistic surgical phantom, shown in Fig. 6, developed by ACMIT Gmbh. It is composed of a 3D-printed pelvis bone and the following silicone elements: rectum, bladder, urethra, seminal vesicles, vas deferens, and prostate, all immersed in a silicate fat-like foam. On these realistically-reproduced anatomical structures we operate with the first model of the da Vinci surgical robot (since 2000) controlled via the daVinci Research Kit (dVRK) and two Franka Emika's Panda robots with custom laparoscopic adapters, designed by Mr. Matteo Piano, as end-effector (SARAS1/SARAS2). The right/left instrument arms (PSM1/PSM2) and the endoscope arm (ECM) are teleoperated by the surgeon, whereas the Panda robots are autonomous. This model of da Vinci does not provide a fourth arm (Fig. 6). Before proceeding with the execution of the surgical procedure, all the robotic arms and the vision system are registered together to create a common reference frame following the methodology proposed in [33]. Moreover, the operative scene is reconstructed in 3D to identify pre-operative points of interest that are mapped to a live point cloud reconstruction during the operation [34]. The latter allows to precisely track all exogenous actors, for instance the catheter. The details of the solutions adopted for scene reconstruction and mapping can be found in the supplementary material attached to this article.

To validate the proposed architecture, we invited three surgeons with experience in robotic-assisted radical prostatectomy to attempt the procedure in cooperation with the SARAS autonomous system. Before the validation, each surgeon has been briefed on the functionality of the system with an explanation on the speech commands list, the way the autonomous system detects the catheter, and the instructions on the user interface available to them in the da Vinci master console; the whole training took approximately twenty minutes. In total, the validation is composed of 38 executions of SARAS-related phases of the RARP, including 10 repetitions of the *bladder mobilization*, 14 of the *bladder neck transection*, and 14 of the *vesicourethral anastomosis*. In the following paragraphs, we proceed with an in-depth analysis of the bladder neck transection phase due to its comprehensive application of all the surgemes and its extensive interaction requirements with the main surgeon. In Fig. 7 a sequence of snapshots shows the main steps of the phase and in Fig. 8 a sequence diagram shows the temporal relation between transitions.

The procedure starts in the *idle* phase with the configuration shown in Fig. 7. The robot is still in a safe position waiting for the surgeon's speech command to trigger the *bladder neck transection* phase. When the command is received (t^l_{16}), the system transits to the *bladder neck transection* phase. As consequence also the current action and surgeme are modified, which are set respectively to *Follow* and S^l_1, as shown in Fig. 7. In the meantime, the *catheter observer* is working concurrently, searching for the presence of the catheter.

Once the catheter is found (t^l_3) the current surgeme transits to S^l_2, and the autonomous system moves the end effector at 3.5 cm apart from the catheter position as shown in Fig. 7. The main surgeon starts the catheter extraction by grasping and pulling upward the catheter with the intent of placing the catheter tip in a position that facilitates the grasping by the autonomous system. In the meantime, the *feature observer* analyzes the catheter velocity profile with respect to the threshold and action transitions, Fig. 9.

The velocity exceeds the threshold for a sufficient time at 28.01 s and then drops to a value approximately close to 0 at 28.61 s. This behavior makes the *feature observer* to trigger t^l_6, thus the transition from *follow* to *grasp* is executed. Consequently, the current state of the surgeme is set to S^l_3, as shown in Fig. 7. The autonomous system moves the



(a) Idle
(b) Idle → bladder neck transection, follow, S_1^l
(c) $S_1^l \rightarrow S_2^l$
(d) Follow → grasp, $S_3^l \rightarrow S_4^l \rightarrow S_5^l \rightarrow S_6^l$
(e) $S_6^l \rightarrow S_7^l \rightarrow S_8^l \rightarrow S_9^l$
(f) Grasp → grasp, $S_3^l \rightarrow S_4^l \rightarrow S_5^l \rightarrow S_6^l$
(g) $S_6^l \rightarrow S_7^l$
(h) $S_7^l \rightarrow S_8^l \rightarrow S_9^l$
(i) S_9^l
(j) Grasp → pull, $S_{10}^l \rightarrow S_{11}^l$
(k) $S_{11}^l \rightarrow S_{12}^l \rightarrow S_{11}^l$
(l) $S_{11}^l \rightarrow S_{10}^l \rightarrow S_{11}^l$

Fig. 7. A series of snapshots taken from the experimental validation during the *bladder neck transection*. State transitions are indicated in the captions as arrows and since some of them occur quickly the transitions are sometimes grouped together in the same snapshot. A detailed view of the transitions occurrences is available in Fig. 8.

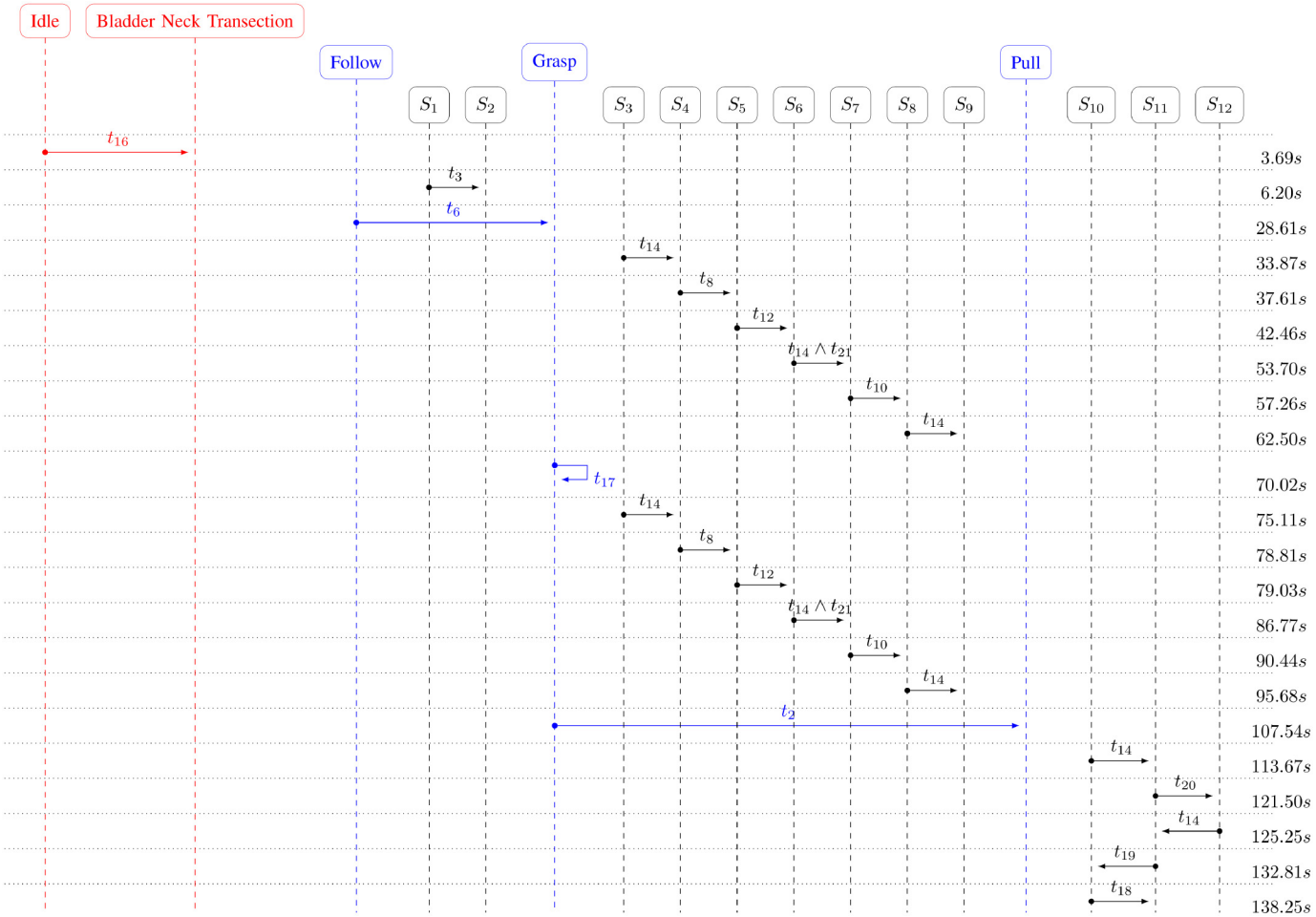
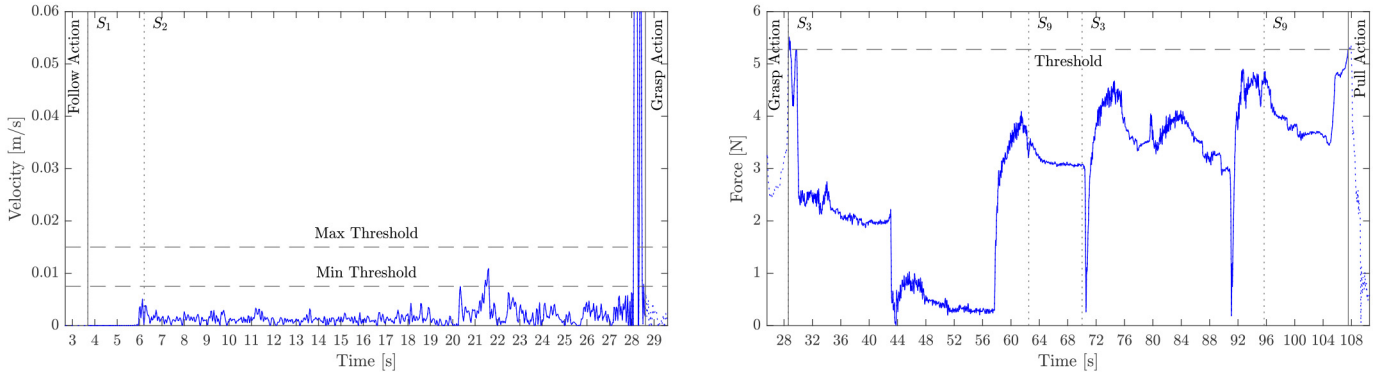


Fig. 8. The sequence diagram shows the time when each transition occurred during the *bladder neck transection*. The time flow is represented by the vertical dashed lines. A transition that occurred between two states is indicated as an arrow, the label above the arrow indicates the trigger which activates the transition. On the right side of the diagram, the transition times are reported. The red, blue, and black colours are used to group the transitions occurring for phases, actions and surgemes respectively.

end-effectors of SARAS 1 and 2 at a 1 cm distance from the catheter (S_3^l); then it rotates the tool (S_4^l), opens the grasper (S_5^l), and moves again the tool at the grasping point (S_6^l). The next transition guards are waiting for the triggers t_{14}^l and t_{21}^l: the former is given when the grasping point position is reached, while the latter is provided by the *speech observer*



(a) (b)

Fig. 9. The catheter speed and the modulus of the interaction force during the *follow* and *grasp* action respectively. a) the horizontal black dashed lines represent the *feature observer* threshold, the vertical black lines represent the action transition and the vertical dotted black lines represent the surgeme transitions, b) the horizontal black dashed line represents the *force observer* threshold, the vertical black lines represent the action transitions and the vertical dotted black lines represent the surgeme transitions.

when the "left arm close the instrument" command is provided. When t^l_{14} and t^l_{21} are triggered, the surgeme S_7 closes the gripper, S^l_8 moves the end effector to verify the grasping, which is followed by S^l_9 which puts the statechart in an idle state waiting for the confirmation to continue. As shown in Fig. 7, during this attempt, the gripper missed the catheter. For this reason, the surgeon triggers the transition t^l_{17}, with the "left arm retry" speech command, as shown in Fig. 7, bringing back the current state to S^l_3. The execution is the same as before until the current state reaches S^l_6: this time the grasping is successful as shown in Fig. 7 and the autonomous system reaches S^l_9 extracting the catheter as shown in Fig. 7. The surgeon pulls the catheter in order to generate a force on the autonomous arm which is monitored by the *force observer*. If the force reaches the desired threshold, the grasping is strong enough to proceed with the next phase of the procedure. Fig. 9 shows the force profile.

Since the monitoring is performed by directly sensing the torques applied at the joints, these measurements could exceed the threshold multiple times during the motion execution due to the motor's controller action. For this reason, the transition could occur only when the surgeme state is S^l_9, which guarantees that the robot has already executed every step necessary to grasp the catheter and it is still. When the force exceeds the threshold (t^l_2), the current action moves to *Pull*, and the actual surgeme is set to S^l_{10}. Then the end effector is moved to the pulling point and when the goal is reached the surgeme t^l_{14} switches the current state to S^l_{11} as shown in Fig. 7. The surgeon requests "left arm pull less" (t^l_{20}) with the specific speech command which sets the current state to S^l_{12} and the end effector is moved near the bladder neck position as shown in Fig. 7. When the position is reached (t^l_{14}) the system gets back to the surgeme S^l_{11}. The catheter is retracted by the autonomous arm according to the main surgeon's instructions. At this point, the surgeon imparts the speech commands to pull the catheter and to stop the motion when the end-effector reaches the desired position. This corresponds to the transition from S^l_{10} to S^l_{11}, and then back to S^l_{10} again. The catheter extraction is so concluded with success.

VI. Discussion and Conclusions

We have evaluated the developed system for autonomous RARP in terms of *phase success rate*, which is expressed as the number of succeeded phases on the total number of executions. We consider to be a successful phase whenever the surgeon is able to accomplish the goal without any fatal error (e.g., damages to the robots or the patient). Any time the surgeon has encountered minor issues (e.g., missing the thread or the catheter grasping), but they has been able to recover the phase without any external intervention, the same is considered successful.

The first phase, the *bladder mobilization*, has minimal interaction with the surgeon, and its success is closely linked to the calibration of the robot with respect to the environment. For this reason, it has required a minimal amount of tests to accomplish the validation, with all of the performed test being successful. No minor nor fatal issues have been encountered in testing this phase.

The validation of the second phase, the *bladder neck transection*, in only one test out of the 14 executed has encountered a fatal error which prevented the phase completion. The issue was caused by an erroneous value of the force threshold which let the statechart unable to transit on the trigger t_2 to the *Pull* action. This was caused by a technical mishap preventing the run-time threshold adaptation and the following procedure interruption. This phase also brought out several minor inconveniences, such as few collisions with the da Vinci arms, minor collisions with the bladder during the movement of the SARAS2 arm and misinterpreted speech commands promptly recovered by the surgeons' direct intervention.

Finally, during the *vesicourethral anastomosis* phase validation, the system encountered a fatal error out of 14 executions. The issue was caused by the surgeon, since he/she failed to correctly point the desired needle approach position causing the SARAS1 arm to start moving toward an undesired and non-reachable configuration. This caused the robot to halt in a non-reachable position that lead to a control system lock and a fatal error. Another issue we faced during this last phase was the SARAS1 arm failure to accomplish the needle's thread

cut. This can be caused by erroneous indications from the main surgeon mainly due to the low resolution of the images displayed on the da Vinci console available with the dVRK platform. These failures, however, only led to the repetition of the positioning movement of the arm near the thread, but they did not prevent the phase completion.

In conclusion, the tests on the platform saw only 2 phases out of 38 interrupted by an unrecoverable error, with both of the issues caused by inappropriate initial positioning of the pre-operative points of interest and the surgeon inexperience while collaborating with SARAS arms. The framework proved to be robust, modular, and flexible by executing successfully the tasks usually performed by an assistant surgeon in a semi-autonomous way. Its design allowed to be quickly extended to heterogeneous tasks while remaining coherent with the formalism already in place for surgical processes. As future works, the authors intend to introduce more nuanced observers that adopt more powerful and adaptable sensing modules based on ML techniques, like the methods presented in [35], along with the development of a proper training session for surgeons to improve usage readiness while maintaining the highest level of native interaction for expert RARP surgeons.

REFERENCES

[1] G.-Z. Yang *et al.*, "Medical robotics—Regulatory, ethical, and legal considerations for increasing levels of autonomy," *Sci. Robot.*, vol. 2, no. 4, p. 8638, 2017.

[2] M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, and P. Fiorini, "Autonomous task planning and situation awareness in robotic surgery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Las Vegas, NV, USA, 2020, pp. 3144–3150.

[3] A. Sozzi, M. Bonfè, S. Farsoni, G. D. Rossi, and R. Muradore, "Dynamic motion planning for autonomous assistive surgical robots," *Electronics*, vol. 8, no. 9, p. 957, 2019.

[4] A. Rector and J. Rogers, "Ontological and practical issues in using a description logic to represent medical concept systems: Experience from GALEN," in *Reasoning Web International Summer School*. Heidelberg, Germany: Springer, 2006, pp. 197–231.

[5] T. Haidegger, "Autonomy for surgical robots: Concepts and paradigms," *IEEE Trans. Med. Robot. Bionics*, vol. 1, no. 2, pp. 65–76, May 2019.

[6] L. Longo, "Argumentation for knowledge representation, conflict resolution, defeasible inference and its integration with machine learning," in *Machine Learning for Health Informatics*. Cham, Switzerland: Springer, 2016, pp. 183–208.

[7] D. Harel and A. Pnueli, "On the development of reactive systems," in *Logics and Models of Concurrent Systems*. New York, NY, USA: Springer, 1985, pp. 477–498.

[8] S. Van Mierlo and H. Vangheluwe, "Statecharts: A formalism to model, simulate and synthesize reactive and autonomous timed systems," in *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems*. Cham, Switzerland: Springer, 2020, pp. 155–176.

[9] Q. Zhao and B. H. Krogh, "Formal verification of statecharts using finite-state model checkers," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 5, pp. 943–950, Sep. 2006.

[10] B. Sobolev, D. Harel, C. Vasilakis, and A. Levy, "Using the statecharts paradigm for simulation of patient flow in surgical care," *Health Care Manage. Sci.*, vol. 11, no. 1, pp. 79–86, 2008.

[11] F. Lalys and P. Jannin, "Surgical process modelling: A review," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 9, no. 3, pp. 495–511, 2014.

[12] N. Djuric *et al.*, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 2095–2104.

[13] S. Zheng, J. J. Lu, N. Ghasemzadeh, S. S. Hayek, A. A. Quyyumi, and F. Wang, "Effective information extraction framework for heterogeneous clinical reports using online machine learning and controlled vocabularies," *JMIR Med. Informat.*, vol. 5, no. 2, p. e12, 2017.

[14] G. Forestier, F. Petitjean, P. Senin, L. Riffaud, P.-L. Henaux, and P. Jannin, "Finding discriminative and interpretable patterns in sequences of surgical activities," *Artif. Intell. Med.*, vol. 82, pp. 11–19, Oct. 2017.

[15] O. Dergachyova, D. Bouget, A. Huaulmé, X. Morandi, and P. Jannin, "Automatic data-driven real-time segmentation and recognition of surgical workflow," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 11, no. 6, pp. 1081–1089, 2016.

[16] A. Murali *et al.*, "Learning by observation for surgical subtasks: Multilateral cutting of 3D viscoelastic and 2D orthotropic tissue phantoms," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, USA, Jun. 2015, pp. 1202–1209.

[17] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg, "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, 2016, pp. 4178–4185.

[18] T. D. Nagy and T. Haidegger, "A DVRK-based framework for surgical subtask automation," *Acta Polytechnica Hungarica*, vol. 16, no. 8, pp. 61–78, 2019.

[19] M. Hwang, D. Seita, B. Thananjeyan, and J. Ichnowski, "Applying depth-sensing to automated surgical manipulation with a da vinci robot," in *Proc. Int. Symp. Med. Robot. (ISMR)*, 2020, pp. 22–29.

[20] A. Kurt and Ü. Özgüner, "Hierarchical finite state machines for autonomous mobile systems," *Control Eng. Pract.*, vol. 21, no. 2, pp. 184–194, 2013.

[21] M. Colledanchise and P. Ögren, "How behavior trees modularize hybrid control systems and generalize sequential behavior compositions, the subsumption architecture, and decision trees," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 372–389, Apr. 2017.

[22] D. Hu, Y. Gong, B. Hannaford, and E. J. Seibel, "Semi-autonomous simulated brain tumor ablation with RAVENII surgical robot using behavior tree," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, 2015, pp. 3868–3875.

[23] R. Muradore *et al.*, "Development of a cognitive robotic system for simple surgical tasks," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 4, p. 37, 2015.

[24] M. Ginesi, D. Meli, H. Nakawala, A. Roberti, and P. Fiorini, "A knowledge-based framework for task automation in surgery," in *Proc. 19th Int. Conf. Adv. Robot. (ICAR)*, Belo Horizonte, Brazil, 2019, pp. 37–42.

[25] Z. Dogmus, G. Gezici, V. Patoglu, and E. Erdem, "Developing and maintaining an ontology for rehabilitation robotics," in *Proc. KEOD*, Barcelona, Spain, 2012, pp. 389–395.

[26] C. E. Reiley and G. D. Hager, *Task Versus Subtask Surgical Skill Evaluation of Robotic Minimally Invasive Surgery* (Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 5761)). Heidelberg, Germany: Springer, 2009, pp. 435–442.

[27] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Amsterdam, The Netherlands: Elsevier, 2004.

[28] C. E. Reiley *et al.*, "Automatic recognition of surgical motions using statistical modeling for capturing variability," in *Medicine Meets Virtual Reality 16 : Parallel, Combinatorial, Convergent : NextMed by Design (MMVR)*, vol. 132. Amsterdam, The Netherlands: IOS Press,2008, pp. 396–401.

[29] T. Bylander, "The computational complexity of propositional strips planning," *Artif. Intell.*, vol. 69, nos. 1–2, pp. 165–204, 1994.

[30] V. R. Patel, K. K. Shah, R. K. Thaly, and H. Lavery, "Robotic-assisted laparoscopic radical prostatectomy: The Ohio state university technique," *J. Robot. Surg.*, vol. 1, no. 1, pp. 51–59, 2007.

[31] E. Oleari *et al.*, "Enhancing surgical process modeling for artificial intelligence development in robotics: The SARAS case study for minimally invasive procedures," in *Proc. 13th Int. Symp. Med. Inf. Commun. Technol. (ISMICT)*, Oslo, Norway, 2019, pp. 1–6.

[32] F. Setti *et al.*, "A multirobots teleoperated platform for artificial intelligence training data collection in minimally invasive surgery," in *Proc. Int. Symp. Med. Robot. (ISMR)*, Atlanta, GA, USA, 2019, pp. 1–7.

[33] A. Roberti, N. Piccinelli, D. Meli, R. Muradore, and P. Fiorini, "Improving rigid 3-D calibration for robotic surgery," *IEEE Trans. Med. Robot. Bionics*, vol. 2, no. 4, pp. 569–573, Nov. 2020.

[34] N. Piccinelli *et al.*, "Rigid 3D registration of pre-operative information for semi-autonomous surgery," in *Proc. Int. Symp. Med. Robot. (ISMR)*, Atlanta, GA, USA, 2020, pp. 139–145.

[35] G. De Rossi *et al.*, "A first evaluation of a multi-modal learning system to control surgical assistant robots via action segmentation," *IEEE Trans. Med. Robot. Bionics*, vol. 3, no. 3, pp. 714–724, Aug. 2021.