# SARTRES: A Semi-Autonomous Robot TeleopeRation Environment for Surgery

Md Masudur Rahman[1*], Mythra V. Balakuntala[2*], Glebys Gonzalez[3], Mridul Agarwal[4], Upinder Kaur[2], Vishnunandan L. N. Venkatesh[2], Natalia Sanchez-Tamayo[3], Yexiang Xue[1], Richard M. Voyles[2], Vaneet Aggarwal[3,4], and Juan Wachs[3]

[1]Department of Computer Science, Purdue University, West Lafayette, IN, 47907, USA . Email: rahman64, yexiang@purdue.edu
[2]School of Engineering Technology, Purdue University, West Lafayette, IN, 47907, USA. Email: mbalakun, kauru, lvenkate, rvoyles@purdue.edu
[3]School of Industrial Engineering, Purdue University, West Lafayette, IN, 47907, USA. Email: gonza337, sanch174, jpwachs@purdue.edu
[4]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA. Email: agarw180, vaneet@purdue.edu

**ARTICLE HISTORY**
Compiled February 28, 2021

**ABSTRACT**

Teleoperated surgical robots can provide immediate medical assistance in austere and hostile environments. However, such scenarios are time-sensitive and thus, require high-bandwidth and low-latency communication links which might be unavailable. Systems with a higher degree of autonomy can address these issues as they can operate even with intermittent feedback from the surgeon. The system presented in this paper has a standard surgical teleoperation interface, which provides surgeons with an environment on which they are trained. In our semi-autonomous robotic framework, high level instructions are inferred from the surgeon's actions and then executed semi-autonomously on the robot. The framework consists of two main modules: (i) Recognition Module - which recognizes atomic sub-tasks (i.e., surgemes) performed at the operator end, and (ii) Execution Module - which executes the identified surgemes at the robot end using task contextual information. The peg transfer task was selected for this paper due to its importance in laparoscopic surgical training. The experiments were performed on the DESK surgical dataset to show the effectiveness of our framework using two metrics: user intervention, measured in terms of degree of autonomy, and success rate of surgeme execution. We achieved an average accuracy of 91.5% for surgeme recognition and a success rate of 86% during surgeme execution. Furthermore, we obtained an average success rate of 53.9% for the overall task, using a model-based approach with a degree of autonomy of 99.33%.

**KEYWORDS**
Teleoperated Robotic Surgery; Surgical activity recognition; Surgical vision and perception

## 1. Introduction

Austere environments, such as battlefields and rural areas, are often lacking in terms of emergency care and life-supporting infrastructure. In dire situations where there is a potential for

* These authors contribute equally to this work

loss of life or grave injuries, immediate medical intervention is required for reducing the number of casualties. However, medical care providers are often lacking in numbers in such environments. Further, in case of a battlefield, the medics are often the first targets. Hence, in such cases, autonomous robotic teleoperation can provide support for treatment of causalities.

While fully autonomous surgical systems are still out of reach today, a semi-autonomous teleoperated robotic medical assistant is a viable predecessor. We envision a system which creates semi-autonomy by sharing task execution between the robot and the human surgeon in teleoperation and utilizes this to circumvent communication bottlenecks. (Garcia et al. 2009; Newman et al. 2011) are examples of robotic teleoperation systems that provide timely medical care to patients with life-threatening injuries. Both systems use direct control teleoperation where continuous multimodal data streams (i.e. video streams, robot kinematics, etc.) are communicated from the operator's side to the robot. Such systems with direct control are not practical in austere environments due to the limitations of such settings, such as: high latency, limited transmission rates, intermittent connectivity, and long-distance communication leading to delays (Loyall et al. 2012; Kay and Thorpe 1995). In these scenarios, a higher level of autonomy is desirable. (Hu et al. 2018; Opfermann et al. 2017) showed that semi-autonomous surgical systems with limited human inputs could perform complicated tasks. (Hu et al. 2015) proposed a semi-autonomous system for tumor ablation which presents the operator with a set of possible execution plans. Once a plan is selected, the robot performs the task autonomously. (Opfermann et al. 2017) developed a system for tumor resection where the user is initially required to identify the extraction points of the tumor, then this information is used to execute the task autonomously. These previous works demonstrate the ability of semi-autonomous systems in completing surgical tasks with limited human intervention.

We propose a framework that can complete the tasks, using a fraction of the data needed in regular teleoperation, but does not require the user to explicitly input their high level decisions. Instead, the system infers the sub-tasks that the surgeon performed and encodes it into high-level abstractions which are then communicated to the robot. In order to do this, we employ a task decomposition strategy which comprehends the execution in terms of atomic subtasks that are refereed to as surgemes (Lin et al. 2006a). Our system infers these surgemes from the low-level kinematic and visual data, and then executes them on the robot. The framework was tested on an ABB Yumi robot during a peg transfer task. The peg transfer task is one of the five tasks in laparoscopic training included in the curriculum, Fundamentals of Laparoscopic Surgery (Ritter and Scott 2007) and is used to train surgeons on bi-manual handling of small objects. Furthermore, to quantify the semi-autonomous capability of our system, we use the metric of the degree of autonomy of the system in terms of the information required from the user to complete the task. Higher the degree of autonomy, lesser the amount of information the system needs from the operator.

## 2. Background and related work

Teleoperation is the fundamental ability to control a robot from afar, using sensing modalities to enable feedback and allow proper situation awareness. Taxonomies describing the different type of teleoperation systems are available in the robotics literature, and the reader can refer to review papers, such as (Adamides et al. 2014), summarizing the main works according to their performance, task, level of autonomy, etc. In our work, we are concerned with two main categories for such taxonomy: 1) Direct control via closed-loop feedback and 2) High level control via simulated feedback (Lichiardopol 2007; De Barros and Linderman 2009).

In direct control via closed-loop feedback, the operator manipulates a robot using continuous feedback received from the remote site (Haidegger et al. 2011; Kofman et al. 2005).

This feedback is streamed either in the form of RGB videos or with kinematic or haptic data (Ferland et al. 2009; You et al. 2018). Hence, these systems are not suitable for real-time operation under communication delays and intermittent connectivity (Kay and Thorpe 1995; Frank et al. 1988).

In contrast, high-level control via simulated feedback involves the operator interacting with a simulated version of the robot that is deployed at the remote end. The remote robot receives high-level instructions from the user and delivers state information back to the user side. This form of control requires accurately modeling the remote scene on the simulator side to provide real-time situation awareness (Douissard et al. 2019). Our work falls in the second category of teleoperation, where operators can interact only with the simulated model and only high-level instructions are sent to the remote robot. Since our work focuses on telesurgery, next we discuss teleoperation in surgery.

Teleoperated robots in surgical settings can be categorized into one of the four levels of autonomy (Yip and Das 2017): (1) pure teleoperation, where the robot just mimics the surgeon's motions; (2) shared teleoperation, where the robot and human share parts of the surgical task and work in tandem (Taylor et al. 1999); (3) semi-autonomous teleoperation, where the surgeon holds control and the robot receives high level information from the surgeon and executes the task; and (4) purely autonomous, where the robot replaces the role of the surgeon completely. The da Vinci robot (Guthart and Salisbury 2000) and the Raven robot (Rosen et al. 2011) are examples of robots that are purely teleoperated by surgeons. Even though there are not fully autonomous surgical systems, partial autonomy has been observed in debridement (Kehoe et al. 2014) and suturing (Pedram et al. 2017) tasks. Our framework provides a reliable middle ground in autonomy by leveraging the expertise of the surgeon together with semi-autonomous execution on the robot side. The surgical actions of the surgeon are recognized through learned models and sent to the robot in the form of high level instructions referred to as surgemes (Lin et al. 2006b).

Surgemes as a fundamental unit of surgical activity has been first proposed by (Lin et al. 2006a). Later in (Reiley and Hager 2009) surgemes were used for quantifying clinical skills among surgeons and even surgical proficiency. By observing the surgemes, it was possible to identify a novice from an expert surgeon. Further, surgemes were also used for task decomposition. When a surgical procedure is decomposed into surgemes (with proper parametrization), they can be utilized to recognize patterns to create semi-autonomous robotic systems (Sen et al. 2016). To create such systems, surgeme recognition must occur first (Reiley and Hager 2009).

Consequently, a short review of the most common methods for surgeme classification include Hidden Markov Models (Reiley and Hager 2009), Conditional Random Fields (Ahmidi et al. 2017), bag of spatio-temporal features (Ahmidi et al. 2017) and Recurrent Neural Networks (DiPietro et al. 2016). Our approach combines kinematic and visual features to generate surgemes for remote execution of the task.

## 3. Methodology

The described framework encodes the kinematic and video data from the operator side into surgeme labels and creates high-level commands for surgical task execution. These commands are then sent through a communication network to the remote robot, which then executes them using a model-based approach.

During data collection, the user teleoperated the robot using a two-handed gaming controller. The controllers allow high motion resolution (1 mm for displacement tracking and 1 degree for rotation tracking) in six degrees of freedom. Specifically, the SRI Taurus was oper-

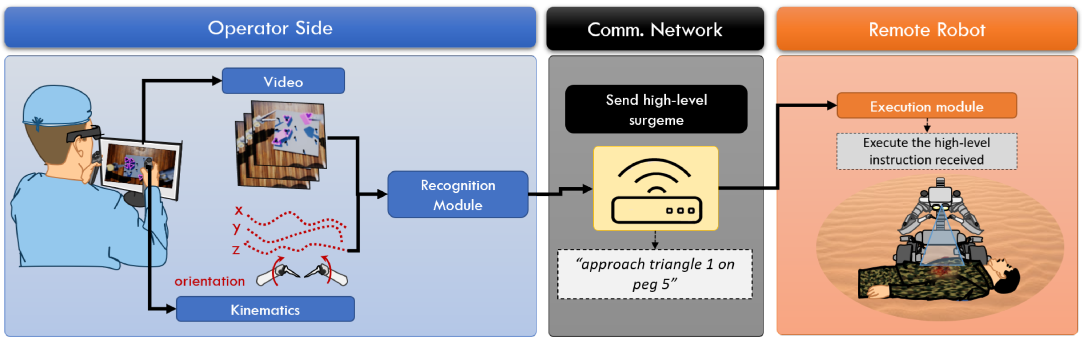

Figure 1.: An overview of the SARTRES framework. The multi-modal recognition module encodes recognized surgemes into high-level commands which the execution module executes.

ated using the Razer Hydra™, the simulated Taurus used the Oculus Rift VR system™, and the YuMi robot was teleoperated through the HTC Vive™. The SRI Taurus is equipped with a stereo camera that is streamed to the user on a 3D monitor. In contrast, the simulated robot feedback was shown directly through the Oculus Rift headset. Finally, the Yumi robot was manipulated through direct observation. For each trial, RGB-D video of the scene, along with the robot's kinematics were recorded.

This framework is subdivided into two main modules for implementation: (1) The Recognition Module, which recognizes the high-level actions (surgemes) performed by the operator, and (2) the Execution Module - which autonomously executes surgeme commands. Figure 1 shows an overview of our architecture.

### *3.1. Recognition Module*

Our approach combines features extracted from video frames and kinematic data to perform surgeme classification based on the multi-modal recognition system proposed in (Rahman et al. 2019).

First, the images extracted from video frames were resized from $1920 \times 1080$ to $228 \times 128$ size. This resized image is then passed to a ResNet18 (He et al. 2016) (pre-trained on ImageNet) to extract image features. Due to the large dimension of the output of the truncated ResNet18, a PCA model is used to reduce the dimension of the features to a lower-dimensional ($= 30$) vector. Additionally, we extracted features from the kinematic data by encoding the features into a shared representation. This representation comprises of the position, orientation and gripper status of the end-effectors (14 features: seven features for each arm) (Madapana et al. 2019).

The surgeme instances (frames of both kinematic and video) were re-sampled to a fixed number of frames (40) to generate sequential feature instances for each surgeme using linear interpolation. The 14 kinematic features for each frame were then concatenated into a single 560-dimensional ($40 \times 14$) vector. Similarly, for image frames, 30 features were concatenated resulting in a 1200 dimensional ($40 \times 30$) feature vector for each surgeme instance. The combined video features were further reduced by PCA to a 100-dimensional feature vector for the entire image sequence in that time-window.

The visual and kinematic features were used to train two separate supervised learning algorithms. One classifier is trained using the visual features and the other classifier is trained on the kinematic features. The final class probability is calculated as a combination of the output class probabilities of the two classifiers with a hyper-parameter $\lambda$, as shown in equation 1.

$$P(C) = \lambda P_{kin}(C) + (1-\lambda)P_{video}(C), \tag{1}$$

where $C$ is the class and $P_{kin}(C)$ and $P_{video}(C)$ are the probabilities given by the classifiers using kinematic and video features respectively. For the experiments in this paper, the value of $\lambda$ is set to 0.8. The values for the hyperparameters — PCA dimensions and $\lambda$ were obtain empirically by tuning them using a grid search on the dataset. The hyperparameter values that provided the highest recognition accuracy are used for the experiments.

### *3.2. Execution Module*

The execution module has two components, a visual recognition unit, and a surgeme execution unit. The visual unit performs scene recognition to extract features or parameters relevant to the surgemes such as the object pose and points of interest. Leveraging this information, the execution unit employs a model-based approach to perform the surgemes sent by the recognition module. The overview of the execution module is shown in Figure 2 and the details of the components are discussed below.

**Visual Recognition Unit**. The visual recognition unit uses RGB-D (RGB + depth) images to extract the 3D object pose, robot tool-tip pose and points of interest. These feature are detected using two networks (Darknet (YoloV3) (Redmon and Farhadi 2018), and Mask-RCNN (He et al. 2017)), a tracker and a feature extractor.

The YoloV3 (Darknet) network is used for detection of 2D object bounding boxes in every RGB frame. For the peg transfer task, the pegs, the task objects and the robot tool-tips are recognized from the scene. As multiple items of the same class may exist in the scene, an object tracker is added which uses the outputs of the YoloV3 network. The object tracking algorithm used in this work is an extension of SORT (Bewley et al. 2016). The extensions account for noise, occlusions and failed detections. The noise is eliminated by using a filter to smooth the motion of the objects across frames. Object ID reassignment, due to occlusions or detection failures, is prevented by using average velocity of objects and predicted position from Kalman filter to estimate their positions in the current frame. Once the object is visible in the current scene again, its position is updated by the tracker. False positives are avoided by accumulating detections across last 10 frames to estimate a confidence of object presence.

The points of interest are extracted by first performing instance segmentation to extract the object shapes, followed by feature extraction from the shape masks. Mask-RCNN network, which is based on a Feature Pyramid Network (FPN) and a ResNet101 backbone (Abdulla 2017) is used to achieve instance segmentation. The object bounding boxes from the tracker and masks from Mask-RCNN serve as inputs to the Point of Interest (POI) module which extracts the surgeme parameters on these objects such as grasp points, which are required for surgeme execution (highlighted points in the Surgeme parameters block in Figure 2).

**Surgeme Execution Unit**. The execution module receives high-level surgeme commands which include the label of the surgeme to be preformed and the relevant parameters of the command (i.e. peg number, dominant arm). The execution of surgemes is performed using a model-based approach. Our framework currently executes surgemes for the peg transfer task. The task consists of seven distinct surgemes - approach, align and grasp, lift, transfer-get together, transfer-exchange, approach target and align and place.To execute a particular surgeme two primary inputs are required from the recognition module, the surgeme label and the parameters corresponding to the surgeme (an example of parameters would be goal object for the 'approach' surgeme or the goal peg for the 'align and drop' surgeme). The "approach"

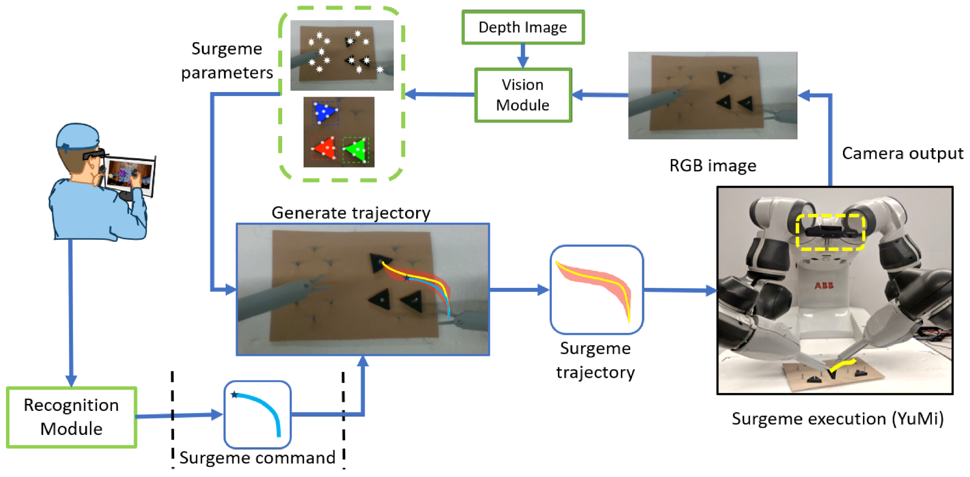

Figure 2.: The architecture of the execution module. The surgeme trajectory block shows the region of possible trajectories (red region) and the actual trajectory the robot would perform (yellow line).

surgeme recognizes the peg of interest according to the received command and extracts a region of interest (ROI) from the output of visual recognition unit. Based on the geometry of the objects, closest area for approach is identified; the tool tip is then moved to this area. In the "align and grasp" surgeme, the robot calculates feasible grasping points. The Point of Interest module uses the masked image of the objects to estimate suitable grasp points. For the "transfer - get together" surgeme, both arms are brought to a middle point of the work space. For the "transfer-exchange" surgeme, the non-dominant hand grasps the object while the initial hand releases it. For the "approach target" surgeme, the center of the object is detected and aligned in the horizontal plane with the target peg. The "align and place" surgeme moves down and releases the object on the target peg.

## 4. Experiments and Results

**Experiment Setup:** The SARTRES framework was evaluated for the peg transfer surgical task. The task consists of a board with multiple pegs, some pegs with objects and some without. The objective of this task is to pick an object (triangular toroid in this case) from one peg using one arm, transfer it to the other arm, and align and place the object on the destination peg. The task poses a challenge especially when dealing with robotic grasps involving obstacle avoidance and maneuverability amongst multiple objects in the task environment.

For recognition module, we used a computing machine with Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, L1d cache: 32K, L1i cache: 32K, L2 cache:1024K, L3 cache: 14080K. The machine contains two nodes and 10 cores per node. We did not use a parallel implementation, thus, our program only uses a single CPU. Memory Per node: 96 GB.

The experiments in this paper were conducted using a subset of the DESK dataset (Madapana et al. 2019). The Taurus simulator S1-S8 (mobile pegboard subjects), Taurus S1-S8, and YuMi S1-S8 data was used for the experiments. In this dataset, the video and kinematic data were segmented manually according to surgemes observed in RGB video frames. Detailed statistics of the surgemes can be found in Table 1. For each robotic domain, 80% of the surgeme data instances were used to train the recognition module and the remaining 20% was used for testing. The testing subset was used to evaluate the performance of the complete teleoperation framework. That is, the videos from the testing fold in the recognition module were used for the execution of experiments. The surgeme execution is conducted on an ABB YuMi robot with surgical extensions. The ABB YuMi robot has 7 DOF for each arm and can be controlled over its joint space or cartesian space. The surgical extensions on the robot comprised of the tool-tips and they were 3D printed. The robot was controlled using the *yumipy* module. This module is a Python interface to commanding the ABB YuMi robot and it is built on the autolab core module. A 3D camera (Intel Realsense) is mounted on top of the ABB YuMi robot, which continuously streams color (RGB) and depth image frames at 30 fps. The convolutional weights for the YoloV3 network (pretrained on Imagenet) were retrained with 5 videos of YuMi data from the DESK dataset with manual annotations for objects. The Mask-RCNN network was also trained using the same YuMi video data, starting from pre-trained weights for COCO dataset. Each of these components were implemented over an overlying ROS network and custom messages and topics were used over ROS for the same. Thus communication between any two elements, such as the MASK-RCNN and the YuMi robot, was carried out seamlessly over ROS. The vision system has a detection accuracy of 98.6% for the objects, 95.2% for the pegs and 96.3% for the tool-tips on a testing set of 290 objects, 587 pegs, and 183 tool segments in 50 images. The execution module took 29.128 seconds on average ($\pm$0.471 seconds) to successfully complete a full peg transfer.

Table 1.: Surgemes and dataset statistics (used in this paper) for the peg transfer task. The columns indicate surgeme ID, name of the surgeme, number of instances of each surgeme in the simulator, the real Taurus and the YuMi robot data.

| ID | Surgeme name | # Sim | # Taurus | # YuMi |
|---|---|---|---|---|
| S1 | Approach triangle | 170 | 129 | 117 |
| S2 | Align & grasp | 171 | 131 | 123 |
| S3 | Lift triangle | 165 | 130 | 123 |
| S4 | Transfer triangle - Get together | 150 | 129 | 117 |
| S5 | Transfer triangle - Exchange | 146 | 129 | 118 |
| S6 | Approach peg | 137 | 127 | 117 |
| S7 | Align & place | 134 | 125 | 116 |

### *4.1. Degree of Autonomy*

In the SARTRES framework, the robot needs reduced user intervention to perform the task. To quantify this capability we measure the degree of autonomy of the robot in this framework. The degree of autonomy was defined by (Barber and Martin 1999) as the degree to which the robot's operation and decision making is free from direct human intervention. In a traditional teleoperation setting, the robot exactly follows the operator's motion, hence having complete intervention. Whereas, for a purely autonomous system, the robot needs no inputs from the operator i.e., there is no intervention from the user. Any semi-autonomous system lies between the aforementioned cases and requires some intervention in the form of high level instructions (surgemes) or coaching inputs.

To measure the degree of autonomy, we propose quantifying the intervention of the surgeon in terms of mutual information, $I(\boldsymbol{X};\boldsymbol{Y})$, between the data provided by the teleoperator ($\boldsymbol{X}$) and the data required by the remote robot ($\boldsymbol{Y}$) to complete the task (Cover and Thomas 2012). Based on the degree of autonomy, the robot receives an abstraction of the data provided by teleoperator, $\boldsymbol{X}$. To execute the task, the robot has to reconstruct the teleoperator data $\boldsymbol{X}$ from the received data $\boldsymbol{Y}$. The reconstruction $\widehat{\boldsymbol{X}}$ may not be perfect. Thus, we calculate the distortion between the input sequence $\boldsymbol{X}$ and the reconstructed sequence $\widehat{\boldsymbol{X}}$, to measure the rate of information that the robot must autonomously reconstruct. The distortion can be defined with respect to a valid distance function $d : \mathscr{X} \times \hat{\mathscr{X}} \to \mathbb{R}^+$, where $\boldsymbol{X}$, and $\hat{\boldsymbol{X}}$ assume their values from the sets $\mathscr{X}$, and $\hat{\mathscr{X}}$ respectively (Cover and Thomas 2012). The distortion is now defined as

$$D(\boldsymbol{X},\widehat{\boldsymbol{X}}) = \mathbb{E}_{\boldsymbol{X}}\left[d(\boldsymbol{X},\widehat{\boldsymbol{X}})\right]. \tag{2}$$

In the context of system autonomy, distance function is the task success metric. The task success metric is usually independent from task performance, which is based on quality metrics such as time to completion or path tortuosity (Fard et al. 2018). To ensure a lateral comparison between two systems with different degrees of autonomy, we add a constraint $D(\boldsymbol{X},\widehat{\boldsymbol{X}}) < D^*$, which implies that the systems have a similar task success rate.

We now compare the degree of autonomy of the robot in the SARTRES framework against a standard teleoperated robot, in terms of the mutual information. In the context of these systems, the input data $\boldsymbol{X}$ is the trajectory followed by the user. The received data $\boldsymbol{Y}$ for the SARTRES framework is the surgeme command from which the trajectories are reconstructed. Whereas for the pure teleoperation system the received data $\boldsymbol{Y}$ is identical to the sent data $\boldsymbol{X}$. The mutual information $I(\boldsymbol{X};\boldsymbol{Y})$ for these systems is,

$$I(\boldsymbol{X};\boldsymbol{Y}) = H(\boldsymbol{Y}) - H(\boldsymbol{Y}|\boldsymbol{X}) \tag{3}$$
$$= H(\boldsymbol{Y}) \tag{4}$$

The last Equation (4) follows from the fact that given a trajectory exactly maps to one surgeme label. Even for teleoperation, as $\boldsymbol{Y} = \boldsymbol{X}$ the mapping is unique. Hence, $P(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \boldsymbol{x}) = 1$ for exactly one $\boldsymbol{y}$, and zero for others and $H(\boldsymbol{Y}|\boldsymbol{X}) = 0$. This result indicates that degree of autonomy is a function of entropy of the transmitted sequence.

Since the sequence $\boldsymbol{Y}$ is not generated by a memory less source, direct calculation of $H(\boldsymbol{Y})$ is non-trivial. From Shannon Source Coding Theorem, we note that, for large sequences, an efficient compression algorithm compresses the sequence to the optimal number of bits required which is equal to the entropy of the sequence (Cover and Thomas 2012). Hence, we measure the entropy $H(Y)$ using the data required to efficiently store $Y$. We use DEFLATE algorithm (Larsson 1996) to compress our data, further, we use gzip implementation for using DEFLATE algorithm.

We define the degree of autonomy of the robot as the percentage of decrease in mutual information with respect to a system with full intervention. For the robot in the SARTRES framework this is,

$$Degree\ of\ Autonomy = 1 - \frac{H(\boldsymbol{Y}_{SARTRES})}{H(\boldsymbol{Y}_{teleoperation})} \tag{5}$$

Measured degree of automation for robot in the SARTRES Framework is presented in Table 2. We note that for distortion of 2% measured as surgeme failure rate, we obtain a mean degree of autonomy of 99.33%.

Table 2.: Degree of Autonomy results for various robot frameworks.

| Robot | Empirical Degree of Autonomy |
|---|---|
| Simulator (Taurus) | 99.57% |
| Taurus | 99.93% |
| Yumi | 98.48% |

## *4.2. Effectiveness of SARTRES*

**Recognition Module**. A supervised learning method, Random Forest (RF), was used (implemented in scikit-learn (Pedregosa et al. 2011)) with hyper-parameters $n\_estimators = 200$ (number of trees in the forest), and $maximum\_depth$ = 10. The combined model's hyper-parameter $\lambda$ was set to 0.8 (empirical best) while combining probabilities from video and kinematic models. The recognition model was trained on three robots separately, namely ABB YuMi, simulated Taurus and SRI Taurus. For each robot, a five-fold cross-validation approach was used with a data split of 80-20% for training and testing respectively.

Table 3 shows the surgeme recognition accuracy using the surgeme Recognition Module from a combination of both kinematics, and video information. The Random Forest model achieves an accuracy of 94.6%, 92.8%, and 87.1% for YuMi, Taurus, and Simulator (Taurus) datasets, respectively.

Table 3.: Classification accuracy of Recognition Module using Random Forest on (Kin+Visual) features.

| Robot | Kin+Visual |
|---|---|
| YuMi | 94.6% |
| Taurus | 92.8% |
| Simulator (Taurus) | 87.1% |

The results for the prediction module are shown in Table 4, discretized according to each specific surgeme. The results are calculated over the 76 complete trials included in the testing set (trials with all S1-S7 surgemes). The results on surgeme prediction are up to 100% for surgemes S1, S2 and S7. Overall, surgeme recognition is on average 91.5% for all surgemes and robotic platforms. The video feature extraction took 1.3 seconds per image and the inference time (using both kinematic and video features) took 0.13 ms per surgeme using the Random Forest classifier (averaged over data for 20 runs).

**Execution Module**. The execution module was evaluated under two different conditions. i) Ground truth scenario, and ii) recognition plus execution (complete framework). The ground truth scenario evaluates execution success rate for the peg transfer task using surgeme ground truth labels obtained from the testing set.

The ground truth scenario evaluates the accuracy of the execution module when the surgemes (high level commands) are received in their totality. In this case, the execution success of surgemes is evaluated only when the previous surgemes of the same trial were successful. In other words, when a surgeme execution fails during testing, the trial stops and no subsequent surgemes in the trial are considered. For this reason, the number of trials evaluated per surgeme decreases for consecutive surgemes. Table 5 shows the performance of the

Table 4.: Per surgeme prediction accuracy of the recognition module on the test data.

| Robot | S1 | S2 | S3 | S4 | S5 | S6 | S7 |
|---|---|---|---|---|---|---|---|
| YuMi (%) | 100 | 96 | 88 | 92 | 96 | 96 | 96 |
| Taurus (%) | 92 | 100 | 88 | 84 | 96 | 92 | 100 |
| Taurus Sim (%) | 100 | 97 | 81 | 90 | 94 | 77 | 73 |
| Average Success (%) | 97 | 98 | 85 | 89 | 95 | 88 | 90 |
| Num Trials | 76 | 76 | 76 | 76 | 76 | 76 | 76 |

surgeme execution module over the ground truth scenario. Results show an average success of 98% per surgeme, for all robotic platforms. Furthermore, the average success of the entire peg transfer trial was 86.6% (all the surgemes in one trial must be performed correctly).

Table 5.: Execution results over ground truth test data.

| Robot | S1 | S2 | S3 | S4 | S5 | S6 | S7 | Task |
|---|---|---|---|---|---|---|---|---|
| YuMi | 100 | 96.0 | 100 | 91.0 | 95.0 | 100 | 100 | 83.0 |
| Taurus | 100 | 96.0 | 100 | 87.0 | 100 | 100 | 100 | 83.0 |
| Taurus Sim | 100 | 96.4 | 100 | 96.3 | 100 | 100 | 100 | 92.8 |
| Average | 100 | 96.1 | 100 | 91.7 | 98.4 | 100 | 100 | 86.6 |
| Num Trials | 76 | 76 | 73 | 73 | 67 | 66 | 66 | 76 |

The complete framework condition evaluates the success of the proposed approach when surgemes are recognized from real kinematic and visual data and evaluated as high level commands on the execution module. Table 6 shows the results and task success rate of the complete framework.

Table 6.: Task completion rate of the proposed framework.

| Robot | Prediction (%) | Task success (%) | Trials |
|---|---|---|---|
| YuMi | 83.3 | 70.8 | 24 |
| Taurus | 66.7 | 64.2 | 24 |
| Taurus Sim | 41.9 | 39.2 | 28 |
| Total | 63.2 | 53.9 | 76 |

We define a *successful trial prediction* as the case where every surgeme is correctly classified in a trial. Similarly, we define *task success* as the accurate recognition and subsequent execution of all surgemes during a trial. Table 6 shows the results for the prediction and execution using these metrics. The successful trial prediction was 83.3%, 66.7%, and 41.9% for the YuMi, Taurus, and Taurus Simulator respectively. Furthermore, the average task success was 53% for all the robots. The highest success rate was achieved with the Yumi robot at 70.8%, since it was the robot with the highest trial prediction success rate.

Finally, The network latency for communication was measured over the Purdue ECN network. Our architecture involves two nodes (robot and the operator) communicating via a central server using TCP sockets. The latency of this setup is 30 ms with a jitter of 25 ms.

### *4.3. Discussion*

In our experiments, the Taurus, the simulated Taurus, and the YuMi robot use different camera positions and orientations. However, we did not explicitly measure the effect of the changing camera location. And, we used same architecture of the recognition module for classifying surgemes from the video data. We have observed that kinematic features give better accuracy compared to video features. A comprehensive analysis of feature importance on this DESK dataset can be found in (Rahman et al. 2019). In our experiments, we find the combined feature (kinematic + video) as the highest performing (accuracy) model, thus we use it with our execution module. The recognition module uses the surgeme frame information for the classification. Thus if the number of frames of a surgeme is higher the module can leverage more information for the detection. For instance, Table 7 shows the surgeme length statistics of Yumi data. We see that surgeme 3 (lift triangle) has the smallest number length (in mean and median). This might cause the classifier to be less accurate for surgeme 3 compared to other surgemes.

| Surgeme ID | Mean # of frame | Median # of frame |
|---|---|---|
| S1 | 133 | 146 |
| S2 | 48 | 62 |
| S3 | **47** | **51** |
| S4 | 216 | 211 |
| S5 | 71 | 79 |
| S6 | 131 | 146 |
| S7 | 84 | 96 |

Table 7.: Surgeme statistics (length - number of frames) on Yumi data for six subjects (S2 - S7).

The experimental setup shows that this version of the system is likely to fail in case of misclassification. Future work shall include using thresholds on the classification probabilities to avoid executing misclassfied surgemes or surgemes with low confidence.

Furthermore, to recover from a state of failure future work should incorporate a feedback module. This module should depict the scene configuration on the local interface, allowing the surgeon to perform recovery actions.

## 5. Conclusions

This paper presents a framework for semi-autonomous teleoperation which reduces human intervention needed to complete the task. The SARTRES framework encodes the user activity into high-level instructions (surgemes) and then remotely executes those surgemes on a robot. In our experiments, the surgemes corresponding to a peg transfer were recognized from videos of three different robotics platforms (real Taurus II, simulated Taurus II and real YuMi) and were executed over the real robot YuMi. To evaluate the degree of autonomy of our system, we present a metric based on the mutual information. With respect to this metric, our system has a mean degree of autonomy of 99.33%.

Our system achieved a surgeme-wise execution accuracy of 98%, task execution success rate of 86.6%, and an overall task success rate of 53.9% for the full framework. These results demonstrate a potential for success of semi-autonomous teleoperated system with high degree of autonomy. Future work can incorporate failure mode recovery into the current architecture to improve the task success rate.

**Acknowledgments**

This work was supported by both the Office of the Assistant Secretary of Defense for Health Affairs under Award No. W81XWH-18-1-0769 and by the NSF Center for Robots and Sensors for the Human Well-Being under Award No. CNS-1439717. Y.X. and M.R. also announce support from the NSF FMitF program (CCF-1918327) and the CR-II program (IIS-1850243). Computational infrastructure was partially supported by Microsoft AI for Earth program. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the funders.

## References

Abdulla W. 2017. Mask r-cnn for object detection and instance segmentation on keras and tensorflow; [https://github.com/matterport/Mask_RCNN].

Adamides G, Christou G, Katsanos C, Xenos M, Hadzilacos T. 2014. Usability guidelines for the design of robot teleoperation: A taxonomy. IEEE Transactions on Human-Machine Systems. 45(2):256–262.

Ahmidi N, Tao L, Sefati S, Gao Y, Lea C, Haro BB, Zappella L, Khudanpur S, Vidal R, Hager GD. 2017. A Dataset and Benchmarks for Segmentation and Recognition of Gestures in Robotic Surgery. IEEE Transactions on Biomedical Engineering. 64(9):2025–2041.

Barber K, Martin C. 1999. Agent autonomy: Specification, measurement, and dynamic adjustment. In: Proceedings of the autonomy control software workshop at autonomous agents; vol. 1999. Citeseer. p. 8–15.

Bewley A, Ge Z, Ott L, Ramos F, Upcroft B. 2016. Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP). p. 3464–3468.

Cover TM, Thomas JA. 2012. Elements of information theory. John Wiley & Sons.

De Barros PG, Linderman RW. 2009. A survey of user interfaces for robot teleoperation.

DiPietro R, Lea C, Malpani A, Ahmidi N, Vedula SS, Lee GI, Lee MR, Hager GD. 2016. Recognizing Surgical Activities with Recurrent Neural Networks. In: Ourselin S, Joskowicz L, Sabuncu MR, Unal G, Wells W, editors. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. Springer International Publishing. p. 551–558. Lecture Notes in Computer Science.

Douissard J, Hagen ME, Morel P. 2019. The da vinci surgical system. Cham: Springer International Publishing. p. 13–27. Available from: https://doi.org/10.1007/978-3-030-17223-7_3.

Fard MJ, Ameri S, Darin Ellis R, Chinnam RB, Pandya AK, Klein MD. 2018. Automated robot-assisted surgical skill evaluation: Predictive analytics approach. The International Journal of Medical Robotics and Computer Assisted Surgery. 14(1):e1850.

Ferland F, Pomerleau F, Le Dinh CT, Michaud F. 2009. Egocentric and exocentric teleoperation interface using real-time, 3d video projection. In: Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction; New York, NY, USA. ACM. p. 37–44. HRI '09; Available from: http://doi.acm.org/10.1145/1514095.1514105.

Frank LH, Casali JG, Wierwille WW. 1988. Effects of visual display and motion system delays on operator performance and uneasiness in a driving simulator. Human Factors. 30(2):201–217.

Garcia P, Rosen J, Kapoor C, Noakes M, Elbert G, Treat M, Ganous T, Hanson M, Manak J, Hasser C, et al. 2009. Trauma pod: a semi-automated telerobotic surgical system. The International Journal of Medical Robotics and Computer Assisted Surgery. 5(2):136–146.

Guthart GS, Salisbury JK. 2000. The intuitive/sup tm/telesurgery system: overview and application. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065); vol. 1. IEEE. p. 618–621.

Haidegger T, Sándor J, Benyó Z. 2011. Surgery in space: the future of robotic telesurgery. Surgical endoscopy. 25(3):681–690.

He K, Gkioxari G, Dollár P, Girshick R. 2017. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. p. 2961–2969.

He K, Zhang X, Ren S, Sun J. 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 770–778.

Hu D, Gong Y, Hannaford B, Seibel EJ. 2015. Semi-autonomous simulated brain tumor ablation with ravenii surgical robot using behavior tree. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE. p. 3868–3875.

Hu D, Gong Y, Seibel EJ, Sekhar LN, Hannaford B. 2018. Semi-autonomous image-guided brain tumour resection using an integrated robotic system: A bench-top study. The International Journal of Medical Robotics and Computer Assisted Surgery. 14(1):e1872.

Kay JS, Thorpe CE. 1995. Operator interface design issues in a low-bandwidth and high-latency vehicle teleoperation system. SAE transactions:487–493.

Kehoe B, Kahn G, Mahler J, Kim J, Lee A, Lee A, Nakagawa K, Patil S, Boyd WD, Abbeel P, et al.

2014. Autonomous multilateral debridement with the Raven surgical robot. In: 2014 IEEE International Conference on Robotics and Automation (ICRA); May. p. 1432–1439.

Kofman J, Wu X, Luu TJ, Verma S. 2005. Teleoperation of a robot manipulator using a vision-based human-robot interface. IEEE transactions on industrial electronics. 52(5):1206–1219.

Larsson NJ. 1996. Extended application of suffix trees to data compression. In: Proceedings of Data Compression Conference-DCC'96. IEEE. p. 190–199.

Lichiardopol S. 2007. A survey on teleoperation. Technische Universitat Eindhoven, DCT report.

Lin HC, Shafran I, Yuh D, Hager GD. 2006a. Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions. Computer Aided Surgery. 11(5):220–230.

Lin HC, Shafran I, Yuh D, Hager GD. 2006b. Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions. Computer Aided Surgery. 11(5):220–230. [accessed 2019-02-24TZ]. Available from: `https://doi.org/10.3109/10929080600989189`.

Loyall J, Gillen M, Cleveland J, Usbeck K, Sterling J, Newkirk R, Kohler R. 2012. Information ubiquity in austere locations. Procedia Computer Science. 10:170–178.

Madapana N, Rahman MM, Sanchez-Tamayo N, Balakuntala MV, Gonzalez G, Bindu JP, Venkatesh LNV, Zhang X, Noguera JB, Low T, et al. 2019. Desk: A robotic activity dataset for dexterous surgical skills transfer to medical robots. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2019).

Newman JG, Kuppersmith RB, O'Malley JB. 2011. Robotics and telesurgery in otolaryngology. Otolaryngologic Clinics of North America. 44(6):1317–31.

Opfermann JD, Leonard S, Decker RS, Uebele NA, Bayne CE, Joshi AS, Krieger A. 2017. Semi-autonomous electrosurgery for tumor resection using a multi-degree of freedom electrosurgical tool and visual servoing. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. p. 3653–3660.

Pedram SA, Ferguson P, Ma J, Dutson E, Rosen J. 2017. Autonomous suturing via surgical robot: An algorithm for optimal selection of needle diameter, shape, and path. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE. p. 2391–2398.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 12:2825–2830.

Rahman MM, Sanchez-Tamayo N, Gonzalez G, Agarwal M, Aggarwal V, Voyles RM, Xue Y, Wachs J. 2019. Transferring dexterous surgical skill knowledge between robots for semi-autonomous teleoperation. In: 28th IEEE International Conference on Robot and Human Interactive Communication (Ro-Man-2019).

Redmon J, Farhadi A. 2018. Yolov3: An incremental improvement. arXiv.

Reiley CE, Hager GD. 2009. Task versus Subtask Surgical Skill Evaluation of Robotic Minimally Invasive Surgery. In: Yang GZ, Hawkes D, Rueckert D, Noble A, Taylor C, editors. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009. Springer Berlin Heidelberg. p. 435–442. Lecture Notes in Computer Science.

Ritter EM, Scott DJ. 2007. Design of a proficiency-based skills training curriculum for the fundamentals of laparoscopic surgery. Surgical innovation. 14(2):107–112.

Rosen J, Hannaford B, Satava RM. 2011. Surgical robotics: systems applications and visions. Springer Science & Business Media.

Sen S, Garg A, Gealy DV, McKinley S, Jen Y, Goldberg K. 2016. Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE. p. 4178–4185.

Taylor R, Jensen P, Whitcomb L, Barnes A, Kumar R, Stoianovici D, Gupta P, Wang Z, Dejuan E, Kavoussi L. 1999. A steady-hand robotic system for microsurgical augmentation. The International Journal of Robotics Research. 18(12):1201–1210.

Yip M, Das N. 2017. Robot autonomy for surgery. arXiv preprint arXiv:170703080:1.

You B, Li J, Ding L, Xu J, Li W, Li K, Gao H. 2018. Semi-autonomous bilateral teleoperation of hexapod robot based on haptic force feedback. Journal of Intelligent & Robotic Systems. 91(3):583–602. Available from: `https://doi.org/10.1007/s10846-017-0738-8`.