Real-Time Context-aware Detection of Unsafe Events in Robot-Assisted Surgery

Mohammad Samin Yasar, Homa Alemzadeh

***Abstract*— Cyber-physical systems for robotic surgery have enabled minimally invasive procedures with increased precision and shorter hospitalization. However, with increasing complexity and connectivity of software and major involvement of human operators in the supervision of surgical robots, there remain significant challenges in ensuring patient safety. This paper presents a safety monitoring system that, given the knowledge of the surgical task being performed by the surgeon, can detect safety-critical events in real-time. Our approach integrates a surgical gesture classifier that infers the operational context from the time-series kinematics data of the robot with a library of erroneous gesture classifiers that given a surgical gesture can detect unsafe events. Our experiments using data from two surgical platforms show that the proposed system can detect unsafe events caused by accidental or malicious faults within an average reaction time window of 1,693 milliseconds and F1 score of 0.88 and human errors within an average reaction time window of 57 milliseconds and F1 score of 0.76.**

I. INTRODUCTION

Robot-assisted surgery (RAS) is now a standard procedure across various surgical specialties, including gynecology, urology and general surgeries. During the last two decades, over 2 million procedures were performed using the Intuitive Surgical's daVinci robot in the U.S. [1]. Surgical robots are complex human-in-the-loop Cyber-Physical Systems (CPS) that enable 3D visualization of surgical field and more precise manipulation of surgical instruments such as scissors, graspers, and electro-cautery inside patient's body. The current generation of surgical robots are not fully autonomous yet. They are in level 0 of autonomy [2], following the commands provided by the surgeons from a master-side tele-operation console in real-time (Figure 1a), translating them into precise movements of robotic arms and instruments, while scaling surgeon's motions and filtering out hand-tremors. By increasing flexibility and precision, surgical robots have enabled new types of surgical procedures and have reduced complication rates and procedure times.

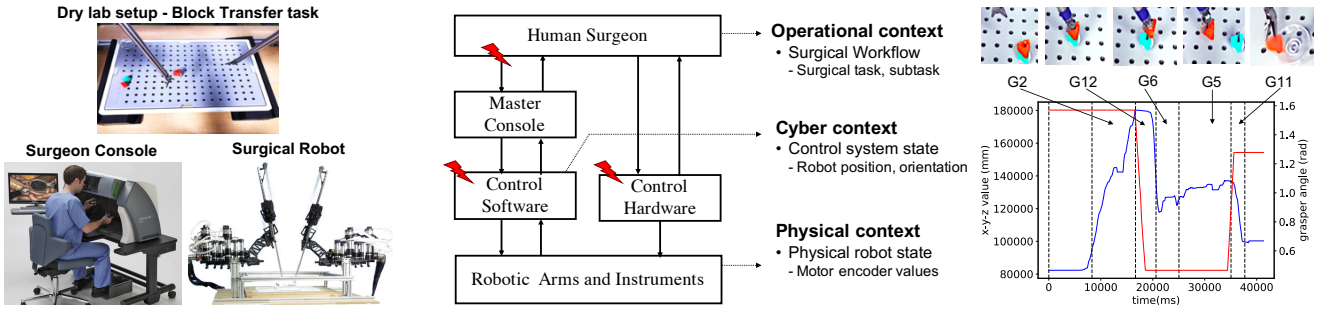
Recent studies have shown that safety in robotic surgery may be compromised by vulnerabilities of the surgical robots to accidental or maliciously-crafted faults in the cyber or physical layers of the control system or human errors [3], [4]. Examples of system faults include disruptions of the communication between the surgeon console and the robot, causing packet drops or delays in tele-operation [5], accidental or malicious faults targeting the robot control software [6], or faulty sensors and actuators [3] (Figure 1b). Those faults can propagate and manifest as system errors (e.g., unintended movements, modification of surgeon's intent, and unresponsive robotic system [7]) or cause human errors (e.g., multiple attempts to suture or end-effector going out of sight [8]–[10]) during surgery and eventually lead to safety-critical events that negatively impact patients and caregivers. Examples include causing unexpected cuts, bleeding, or minor injuries or resulting in long procedure times and other complications during the procedure or afterwards [3]. Table I provides examples of common types of human errors during a sample set of surgical tasks, as reported in the literature.

Our goal is to improve the safety of surgery by enhancing the robot controller with capabilities for real-time detection of early signs of adverse events and preventing them by issuing timely warnings or taking mitigation actions. Previous works on safety and security monitoring and anomaly detection in surgical robots and other CPS have mainly focused on incorporating the information from the cyber and/or physical layers for modeling and inference of the system context [11] and distinguishing between safe and unsafe control commands that could potentially lead to safety hazards. For example, [6] proposed an anomaly detection framework for detection of targeted attacks on the robot control system through modeling robot physical state and dynamics. They showed that considering the physical context (e.g., next motor position and velocities to be executed on the robot) leads to improved detection of unsafe events compared to fixed safety checks on just the cyber state variables (e.g., torque commands calculated in control software). However, with the major involvement of the human operators in real-time control and operation of semi-autonomous CPS such as surgical robots, another important contributing factor to safety is the operational context that captures human operators' preferences, intent, and workflow. In this work, we aim to improve the detection coverage for unsafe events during surgery by considering a more complete view of system context that incorporates the knowledge of the surgical workflow, characterized by the surgical tasks or gestures being performed by the surgeon.

This paper presents an online safety monitoring system for robot-assisted surgery that takes the human operator actions (surgeon's commands at the tele-operation console) as input to infer the operational context (current surgical subtask or gesture) and performs context-specific validation of the kinematics state of the robot to detect erroneous gestures that could potentially lead to safety-critical events. The proposed system can be integrated with the existing surgical robots

*This work was partially supported by a grant from the U.S. National Science Foundation (Award No. 1748737).

*Authors are with the Department of Electrical and Computer Engineering (ECE), University of Virginia, Charlottesville, VA 22904, USA {msy9an, ha4d}@virginia.edu.



(a) (b) (c)

Fig. 1: (a) Tele-operated surgical robot, adopted from [12], [13], (b) Typical control structure and system context in robotic surgery, (c) Operational context characterized by surgical gestures in a sample trajectory of Block Transfer task

and simulators to provide real-time feedback to surgeons and potentially improve the learning curves in simulation training and prevent adverse events in actual surgery. In summary, the main contributions of the paper are as follows:

- Demonstrating that across a surgical task, the errors are context-specific, i.e., dependent on the current surgical gesture being performed by the surgeon (Section II). This suggests the possibility of designing a context-aware monitoring system that can detect gesture-specific errors and can be pervasively applied to any surgical task that is composed of such atomic gestures.
- Developing a safety monitoring system, consisting of a surgical gesture classifier and a library of gesture-specific classifiers that can detect the erroneous gestures in real-time (Section III). Our proposed system can detect errors caused by accidental faults or attacks targeting the software, network or hardware layer or human errors that affect the kinematic state of the robot.
- Introducing a simulation environment based on ROS [14] and Gazebo [15] and the Raven II [12] control software (an open-source surgical robot from Applied Dexterity Inc.), that enables the experimental evaluation of safety monitoring solutions for robotic surgery (Section IV-B). Our simulator can model physical interactions between the robot and its environment, generate synthetic surgical trajectory data, and simulate realistic robot failure modes using software fault injection without harming the physical robot.
- Evaluating our monitoring system in terms of accuracy and timeliness in detecting errors using data from two different surgical platforms, Raven II and daVinci Research Kit [16] (dVRK from Intuitive Surgical Inc.). Our results in Section V provide evidence for our hypothesis that a context-aware safety monitor can enable more accurate detection of anomalies. The proposed monitor can detect potential adverse events for the surgical tasks of Block Transfer and Suturing with average F1 scores of 0.88 and 0.76 within average reaction time windows of 1,693 and 57 milliseconds, respectively.

II. Preliminaries

Our goal is to enhance the surgical robots with capabilities for real-time detection of errors and providing just-in-time feedback to surgeons or taking automated mitigation actions before safety-critical events occur. Our safety monitoring solution is built upon the main concepts described next.

Operational Context in Surgery: The diverse sources of faults and involvement of humans in the control of surgical robots make real-time detection of adverse events particularly challenging and require understanding of the surgical context in order to decide on the best safety actions to take.

Previous work [19] has defined context in surgical procedures as a hierarchy, starting from the surgical procedure that is being executed, to the steps in the procedure, to surgical tasks, subtasks or gestures, and finally the specific motions of the robot (see Figure 2). Within a specific procedure (e.g., Radical Prostatectomy) for a surgical task (e.g., suturing), the change in operational context happens in the temporal domain as a result of the change of the surgeon's gestures or the position and orientation of the surgical instruments end-effectors, leading to the corresponding change in the subtask (e.g., pull suture through). Other contributing factors to the change in the operational context in surgery are the state of the robot software, the type of surgical instrument used, and the anatomical structures and their interactions within the surgical workspace. The operational context can be either observed directly using video data or inferred from the corresponding kinematics data and other state information from the robot.

One common approach to modeling of surgical tasks is using finite-state Markov chains with each state corresponding to an atomic surgical gesture [21]. In our work, we consider tasks from the Fundamentals of Laparoscopic Surgery (FLS) [22], in particular, Suturing and Block Transfer, which are commonly observed in both simulation training and actual surgery. Figure 3a shows the Markov chain representation of the task of Suturing, which we derived from the analysis of the dry-lab demonstrations in the JIGSAWS dataset [23].

Surgical Task	Faults	Errors	Adverse Events
Laparoscopic Cholecystectomy	Wrong orientation of end-effector	Liver laceration with bleeding	Hematoma [17]
Laparoscopic Cholecystectomy	Too much force with grasper	Peroration of gallbladder wall	Subhepatic [17] abcess
Anastomosis	Wrong Cartesian Position	Needle overshoots goal	Punctures [18] Vessel

TABLE I: Common Errors in Surgery

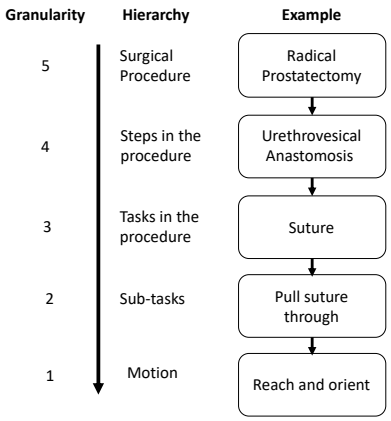


Fig. 2: Hierarchies in Surgical Procedures (adopted from [20])

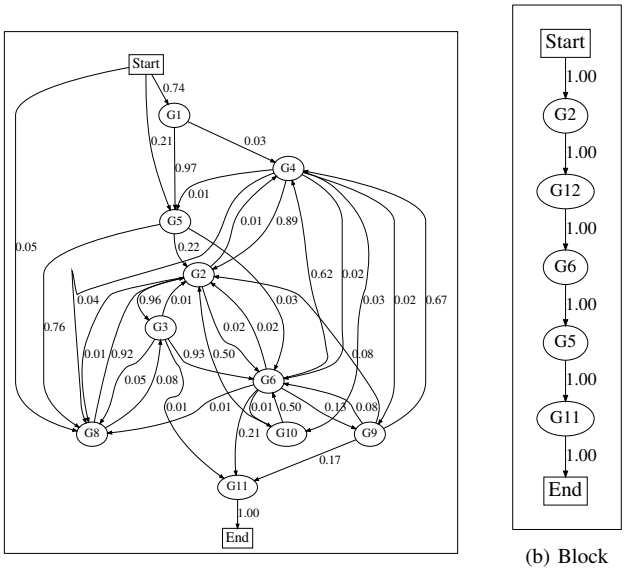
The gestures in Suturing are represented as G1 to G11, excluding G7, as described in Table II. It is apparent that different demonstrations of Suturing could follow different sequences of gestures due to variations in surgeons' styles and common errors in performing the tasks. Figure 1c shows a sample trajectory for the surgical task of Block Transfer, consisting of G2, G5, G6, G11 and G12 gestures. As this is a comparatively simple task, all demonstrations in our collected dataset have the same sequence as seen by the Markov chain in Figure 3b, making the probability of transitioning between different states 1.

Prior works on surgical skill evaluation [24] and safety monitoring [19] have shown that efficiency and safety of surgical tasks are context-specific and that certain gestures or sub-tasks are better indicators of surgeon's skills and surgical outcomes [25]–[27]. We incorporate this concept into our system and provide evidence of improved results if we use the notion of gestures when detecting unsafe events.

Current surgical robots and simulators use surgeon's commands and robot trajectories, collected from surgical procedures or virtual training experiments, for *offline* analysis of subtasks and objective evaluation of surgeon's performance [21], [24], [28]–[30]. In this work, we show that there is potential for the *online* analysis of this data during surgery to prevent the occurrence of safety-critical events.

Erroneous Surgical Gestures: Given the knowledge of surgical gestures, the goal of the safety monitor is to detect erroneous gestures performed on the surgical robot that could indicate the early signs of unsafe events. As there are many variables involved each time a surgical task is performed, starting from the idiosyncrasies of the surgeon (preferences, efficiency, and expertise) to the dynamics of underlying cyber-physical system of the robot, it is safe to assume that there will be many variations of the same surgical gesture. However, it is imperative to identify erroneous gestures that could potentially lead to adverse events or delay in the task. The identification of erroneous gestures as the atomic building blocks of surgical procedures could enable preemptive detection of unsafe events in any surgical task.

We extend the well-established definition of surgical gestures by identifying the common errors that are observed when performing each gesture, using a rubric adopted from [31]. Table II shows the set of gestures and common errors



(a) Suturing (b) Block Transfer

Fig. 3: Markov chain derived for the surgical task of Suturing and Block Transfer

in the tasks of Suturing and Block Transfer. Similar rubrics can be derived for other tasks by identifying their atomic gestures and gesture-specific errors, as shown in [8]–[10], [19], [32]. We classify a gesture as erroneous if any of the common errors specific to that gesture are observed. Not all erroneous gestures will lead to adverse events. Depending on the gesture, the type of error, and other contextual factors, the erroneous gestures can vary in terms of severity but in this work we do not consider this for detecting them.

We also show the types of faults in the kinematic state variables that can potentially cause such errors. We assume that accidental or malicious faults in software, hardware or

Gesture Index	Description	Common Gesture Specific Errors (Failure Modes)		Potential Causes (Faults)
G1	Reaching for needle with right hand	More than one attempt to reach		Wrong rotation angles
G2	Positioning needle	More than one attempt to position		Wrong rotation angles
G3	Pushing needle through the tissue	Driving with more than one movement	Not removing the needle along its curve	Wrong Cartesian Position
G4	Transferring needle from left to right	Unintentional Needle Drop	Needle held on needle holder not in view at all time	Wrong Cartesian Position/Sudden jumps
G5	Moving to center with needle in grip	Unintentional Needle Drop		High Grasper Angle
G6	Pulling suture with left hand	Needle held on needle holder not in view at all times	Unintentional Needle Drop	Wrong Cartesian Position/Sudden jumps
G8	Orienting needle	Uses tissue/ instrument for stability	More than one attempt at orienting	Wrong rotation angles
G9	Using right hand to help tighten suture	Knot left loose		Low pressure applied to tighten suture
G10	Loosening more suture			
G11	Dropping suture and moving to end points	Failure to dropoff		Low Grasper Angle
G12	Reaching for needle with left hand	More than one attempt to reach		Wrong Cartesian Position/Sudden jumps

TABLE II: Gesture specific errors in Suturing and Block Transfer tasks (adopted from [23], [31])

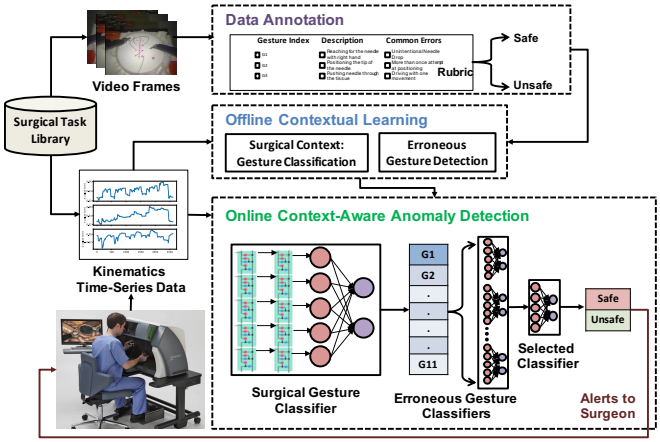


Fig. 4: Pipeline for Real-time Context-aware Safety Monitoring System.

network layer, or human errors can manifest as errors in the kinematics state variables and lead to such erroneous gestures. We demonstrate that this is possible through fault injection experiments on the RAVEN II surgical robot using simulated trajectory data for the task of Block Transfer in Section IV-B and through analysis of the pre-collected trajectory data from dry-lab demonstrations of Suturing on the Intuitive Surgical's daVinci Research Kit in Section IV-A.

III. Context Aware Safety Monitoring

Our context-aware monitoring system is composed of two supervised learning components, 1) A surgical gesture classifier followed by a library of 2) Erroneous gesture classifiers. For both parts of our pipeline, we use variants of Deep Neural Networks (DNNs) [33], which have achieved state-of-the-art performance in many pattern-recognition and data mining problems. We model the detection task as a hierarchical time-series classification problem. The first part of our pipeline is trained to identify the current operational context or gesture. This then activates the second part of the pipeline, which classifies the gesture as safe or unsafe by learning from gesture-specific spatio-temporal patterns in time-series kinematics data. We train the two parts of the pipeline separately from each other. Figure 4 shows our end-to-end pipeline for training and evaluation of the safety monitoring system[1]. The proposed monitor can be integrated with the surgical robot by being deployed on a trusted computing base at the last computational stage in the robot control system pipeline [6] and be used in conjunction with other mechanisms proposed in previous works (see Section VII) to secure the robot against faults and attacks.

Analysis of Erroneous Gesture Distributions: To better understand the characteristics of different erroneous gestures, we performed an analysis of their underlying distributions based on the kinematics data collected from the task of Suturing in the JIGSAWS database. Previous work [19], [34] have modeled the surgical trajectories as a multi-modal Gaussian distribution, with kinematics data being sampled from one of the many Gaussian mixtures and each mixture

[1]Code available at: https://github.com/UVA-DSA/ContextMonitor

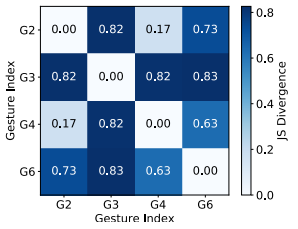


Fig. 5: Pairwise divergence between erroneous gesture distributions

corresponding to a different gesture. We used this insight to estimate the probability density function of each erroneous gesture class using Gaussian kernels. We then calculated the relative entropy between the respective distributions of different erroneous gesture classes (EG_i) using the Jensen-Shannon Divergence (JS-divergence) metric [35] which provides us with a measure of difference between each pair of distributions as calculated in Equation 1:

$$\begin{aligned} JSD(EG_i||EG_j) &= \frac{1}{2}D(EG_i||M) + \frac{1}{2}D(EG_j||M) \\ where,\ M &= \frac{1}{2}(EG_i + EG_j)\ and \\ EG_{i,j} &= Erroneous\ Gestures \end{aligned} \tag{1}$$

Figure 5 shows the pairwise JS-divergence between distributions of different erroneous gestures. We see that there is in particular a high divergence between the distributions of gesture classes G2, G3, G4 and G6, all of which are commonly occurring gestures and have a large number of samples in the task of Suturing (see Table VII). For the other gesture classes we were not able to compute meaningful distributions due to small sample sizes. This observation partly supports our hypothesis that errors in surgery are context-specific and the knowledge of gestures might help with improved error detection. We use this observation in designing our *Erroneous Gesture Detection* component of the pipeline by developing a library of classifiers, each trained for detecting errors in a specific gesture class.

Gesture Segmentation and Classification: We model the task of identifying and segmenting surgical gestures as a multi-class classification problem (Equation 2). The input signal x_t represents the time-series of kinematics variables with a sliding time-window of w and a stride of s. The output G_t represents the gesture corresponding to that time-series and is a one-hot vector of all gestures from 0 to 14.

$$\begin{aligned} x_t &= (x_t, x_{t+1}, .., x_{t+w}) \\ G_t &= (0, 0, 1, .., 0)^T \end{aligned} \tag{2}$$

As the input signal is a multivariate time-series, we use Recurrent Neural Networks (RNN) which are known to learn spatial and temporal patterns. For our gesture classification, we use LSTM [36] networks which are known to learn long and short-term dependencies, and have the ability to decide what part of the previous output to keep or discard. A typical LSTM unit is composed of a memory cell and three gates: input, output and the forget gate. The gates regulate the flow of information inside the unit and allow LSTM architectures to remember information for long periods of time while also filtering information that is less relevant.

To aid our gesture classification and ensure smooth transition boundaries, we use stacked LSTM layers to provide greater abstraction of the input sequence and to allow the hidden states at each level to operate at a different timescale [37]. This is followed by a fully-connected layer with ReLU [38] activation and a final softmax layer for obtaining gesture probabilities. The loss function is the categorical cross-entropy, with the model trained using the Adam [39] optimizer. To address over-fitting, we use dropout regularization and early stopping on a held-out validation set. To improve the learning process, we use batch normalization layers and adaptive learning rate with step-decay.

Erroneous Gesture Detection: Having identified the current gesture, G_t, the next stage of the pipeline classifies the gesture as erroneous or non-erroneous using the kinematics samples as input. We train this part of the pipeline separately from the gesture classification component and only combine the two parts in the evaluation phase, with the erroneous gesture detection following the gesture segmentation and classification (see Section V-B).

$$\begin{aligned} x_t &= (x_t, x_{t+1}, .., x_{t+w}) \\ y_t &= p(EG_t | G_t, x_t) \end{aligned} \tag{3}$$

We frame the problem as detection of a context-specific conditional event, i.e., a part of the trajectory can be erroneous or non-erroneous, depending on the current gesture, as shown in Equation 3. The input is the predicted gesture and a kinematics sample corresponding to that gesture, and the output is a binary classification of the kinematics sample to safe or unsafe. If any sample within a gesture is erroneous, we label that whole gesture as unsafe. Although our model is trained on sliding time-window samples instead of the whole gesture, it learns to have smooth output over time, allowing it to distinguish between entire boundaries of erroneous or non-erroneous gestures.

As a baseline, we trained a single classifier, with no explicit notion of context, for detecting the erroneous gestures. In this case, the problem reduces to a non-conditional binary classification of the time-series data, with the input being the kinematics sample and the output being whether it is erroneous or not. Similar to gesture classification, our models for detecting erroneous gestures are trained using the Adam optimizer with step-decay and early stopping. We used low initial learning rates ranging from 0.0001 to 0.001 to help the stability of the optimization, given a small dataset.

IV. Experiments

We evaluated our monitoring system using trajectory data collected from the common surgical tasks of Block Transfer and Suturing performed in dry-lab settings on two different surgical platforms, the open-source Raven II surgical robot and the daVinci Research Kit (dVRK). The Raven II allowed us to simulate the impact of technical faults and attacks using software fault injection, while the surgical data collected from dVRK enabled studying the effect of human errors and evaluating the safety monitor using realistic surgical tasks.

All experiments were conducted on an x86_64 PC with an Intel Core i7 CPU @ 3.60GHz and 32GB RAM, running Linux Ubuntu 18.04 LTS, and an Nvidia 2080 Ti GPU, running CUDA 10.1. We used Keras [40] API v.2.2.4 on top of TensorFlow [41] v.1.14.0 for training our models and Scikit-learn [42] v.0.21.3 for pre-processing and evaluation.

A. daVinci Research Kit (dVRK)

JIGSAWS Dataset: For evaluating the performance of our solution on the dVRK, we considered the surgical task of Suturing. Since we did not have full access to the system, we used pre-collected trajectory data from the JIGSAWS dataset [23] and manually annotated the errors. The dataset contains synchronized kinematics and video data recorded at 30 Hz from three surgical tasks (Suturing, Knot-tying and Needle-passing) that were performed by surgeons with varying skill levels in dry-lab settings on the dVRK platform. The kinematics data comprises of 19 variables for each robot manipulator, including: Cartesian Position (3), Rotation Matrix (9), Grasper Angle (1), Linear (3) and Angular Velocity (3). We used the data from 39 demonstrations of the task of Suturing with the Leave-One-SuperTrial-Out (LOSO) setup of the JIGSAWS dataset for training and evaluating our monitoring pipeline. The LOSO setup meant that we trained on 4 super trials and held one super trial out for evaluation.

Erroneous Gesture Annotation: The gestures were already labeled as part of the JIGSAWS dataset [23], but we further classified them as safe or unsafe. We did so by manually annotating video data based on the rubric in Table II and used it as ground truth for evaluating our classifiers which rely only on kinematics data. We labeled any given gesture as unsafe if any of the common errors specific to that gesture were observed in its corresponding video segment. Out of a total of 793 gestures, 144 were labeled as erroneous.

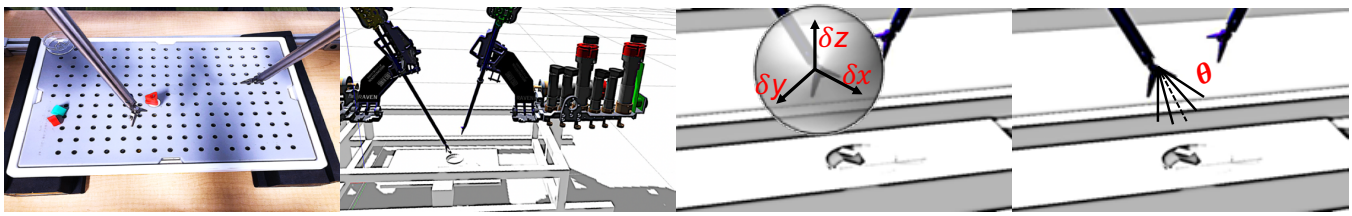
B. Raven II

ROS Gazebo Simulator: For the Raven II, our experiments were conducted using a simulator that we developed based on ROS Gazebo 3D virtual environment, integrated with the RAVEN II control software and a fault injection tool that mimics the effect of technical faults and attacks in the robot control system. This simulator is available to the research community for experimental evaluation of safety monitoring solutions in robotic surgery[2].

We leveraged the physics engine of the Gazebo simulator for faithful representation of the dry-lab settings. Figures 6a and 6b show the dry-lab setup of the Block Transfer task in the Raven II workspace along with the corresponding simulation in the Gazebo 3D environment. All our experiments used the same setup with the left and right robot manipulators, grasper instruments, and the standard objects in the Block Transfer task, including a block and a receptacle where the block should be dropped.

The input to the simulator can be surgeon's commands during tele-operation or output from motion planning algorithms in autonomous mode. The kinematics data from

[2]https://github.com/UVA-DSA/raven2_sim/tree/gazebo_sim



(a) Block Transfer in dry lab setting (b) Block Transfer in ROS Gazebo (c) Cartesian Position Faults (d) Grasper Angle Faults

Fig. 6: Experimental setup for dry lab and virtual simulation of surgical tasks

the simulator consisted of 277 features (including the 19 variables available from the JIGSAWS dataset), sampled at 1000 frames per second. The simulator also allows logging of the video data using a virtual camera. The video frames are logged at 30 frames per second, along with their timestamps to enable synchronization with kinematics data and to measure the times when faults and errors happen. We collected 20 fault-free demonstrations of the Block Transfer task performed by 2 different human subjects in the simulator, on which we carried out our fault injections. The dataset collected from the simulation experiments consisted of 115 fault-free and faulty demonstrations.

Fault Injections: We assume that accidental or malicious attacks and human errors can manifest as errors in the kinematic state variables in the inputs, outputs and the internal state of the robot control software and cause the common error types shown in Table II. As a result, our software fault injection tool directly perturbs the values of kinematic state variables to simulate such errors. Each injected fault is characterized by the name of the state variable (V) with value (S) that is targeted, along with the injected value (S') and the duration of the injection (D).

Figure 1c shows how the operational context for the Block Transfer task changes with respect to the kinematic state variables (V), which are the Grasper Angle and the Cartesian Position of the robot instrument end-effectors. The fault duration, D, is measured in milliseconds and could span across more than one gesture. We perturbed the values of Grasper Angle and the Cartesian Positions (x, y, z) in the collected fault-free trajectories and sent the faulty trajectory packets to the robot control software. This allowed us to repeat the same trajectory or to perturb only specific segments while the rest of the trajectory remained the same.

To simulate the effect of attacks or human errors, we created deviations from the actual trajectory by slight increments or decrements in the values of Grasper Angle and the Cartesian Position variables. For Grasper Angle, we added a constant value of θ for the duration (D) until the target value (S') was reached (see Figure 6d). For Cartesian Position, we provided a target deviation ($\delta = d(S', S)$), which is the Euclidean distance between S and S' and a function of x, y and z values. We then enforced a uniform positive deviation in the three dimensions of x, y, and z by injecting the value of $\delta_{x,y,z} = \delta/\sqrt[2]{3}$ to the three variables over the duration (D) (see Figure 6c).

Table III shows the results of our fault injection experiments on the Gazebo simulator. In total, we conducted 651 fault injections in the task of Block Transfer, out of which, 498 resulted in errors, including 392 block-drop and 106 for dropoff failures. The state-space for injected values, S', was 0.3 rad $\leq S' \leq$ 1.6 rad for Grasper Angle and 3000 mm $\leq S' \leq$ 65000 mm for Cartesian Position. We explored different combinations of perturbations of the targeted variables over different durations of the trajectory.

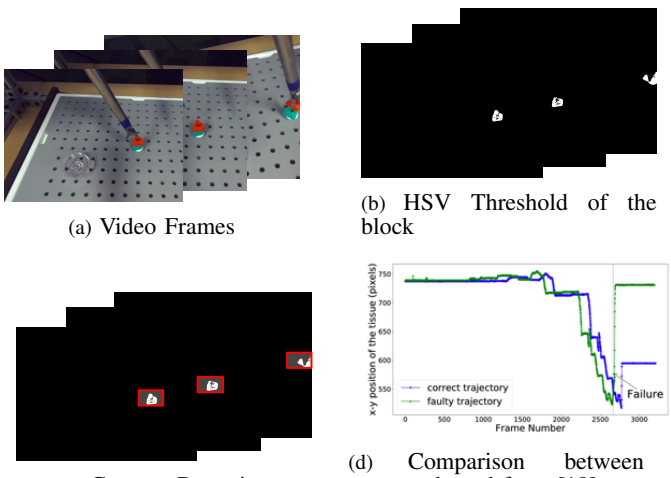
The experiments showed that perturbing the Grasper Angle had a greater effect on causing errors compared to perturbing the Cartesian Position. For lower Grasper Angle values (0.3 rad $\leq$S'$\leq$ 0.8 rad), perturbation over different durations of the trajectory resulted in different failure modes. For fault injections with duration 0.65 $\leq D \leq$ 0.9 and targeted Grasper Angle 0.3 rad $\leq S' \leq$ 0.8 rad, the likelihood of a dropoff failure was high (>90%) whereas for the same range (0.3 rad $\leq S' \leq$ 0.8 rad) but different duration (0.55 $\leq D \leq$ 0.7), the likelihood of any failure was significantly low. There were only 2 cases where the block was dropped at the wrong position due to high Cartesian deviation. When injecting higher values to the Grasper Angle (0.9 rad $\leq S' \leq$ 1.6 rad), we observed block-drops regardless of the duration of the perturbation, with higher values of S' leading to higher percentage of failures. This suggests that for block-drop error to happen, the value of the Grasper Angle either needs to be higher than 0.8 rad, or the fault needs to be injected for a longer duration of time. For dropoff failure to happen, the Grasper Angle has to be below 1.0 rad, or the fault needs to be injected for a longer duration, possibly beyond G11, which is the gesture where the block should be dropped.

Fault Types (Values, Durations, and Total Number)					# Errors (%)	
Grasper Angle (rad)	Duration (% Trajectory)	Cartesian Position Deviation (mm)	Duration (% Trajectory)	# Fault Injections	Block-drop	Dropoff Failure
0.30-0.40	0.55-0.70	3000-6000 6000-65000	0.50-0.60	16 8	0 (0%) 1 (12.50%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16		16 (100%) 16 (100%)
0.50-0.60	0.55-0.70	3000-6000 6000-65000	0.50-0.60	16 8	0 (0%) 1 (12.50%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16		16 (100%) 15 (93.75%)
0.70-0.80	0.55-0.70	3000-6000 6000-65000	0.50-0.60	16 8	0 (0%) 0 (0%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16		15 (93.75%) 16 (100%)
0.90-1.00	0.55-0.70	3000-6000 6000-65000	0.50-0.60	58 50	28 (48.28%) 33 (66%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16	5 (62.50%) 6 (75%)	6 (75%) 6 (75%)
1.10-1.20	0.55-0.70	3000-6000 6000-65000	0.50-0.60	47 74	46 (95.78%) 67 (86.49%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16	12 (75.00%) 12 (75.00%)	
1.30-1.40	0.55-0.70	3000-6000 6000-65000	0.50-0.60	41 61	40 (97.57%) 58 (95.08%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16	14 (87.50%) 14 (87.50%)	
1.50-1.60	0.55-0.70	3000-6000 6000-65000	0.50-0.60	7 17	6 (85.71%) 17 (100%)	
	0.65-0.90	3000-6000 6000-65000	0.70-0.90	16 16	16 (100%) 16 (100%)	
Total Fault Injections				651	392	106

TABLE III: Fault injection experiments on the Raven II

Automated Labeling of Errors: We used computer vision approaches as an orthogonal method of detecting errors, as our fault injections were performed on the kinematics state variables. This, along with the knowledge of when a fault was injected, provided us the semantics to label a particular gesture as erroneous or non-erroneous. We adopted the marker-based (color and contour) detection approaches used in [19] here. As the first step, we converted the logged video data to a sequence of frames (Figure 7a) with their corresponding timestamps. For the case of detecting Block-drop, we used Structural Similarity Index (SSIM) [43] on thresholded images (Figure 7b) of the block to find the exact frame (and the timestamp) of when the failure happened. For the case of detecting Drop-off failure, we applied the same HSV threshold, followed by contour detection (Figure 7c) to detect the contour of the block and track its centroid throughout the trajectory. We collected the trace of the centroid for the fault-free trajectories which we used as reference to compare against faulty trajectories. We used Dynamic Time Warping to compare the fault-free and faulty trajectory traces and checked for large deviations that indicate when the block should have been dropped, but it was not (Figure 7d).

Gesture Annotation: For annotating the data generated using the Gazebo simulator, we extended the data structure of the Raven II to include the current surgical gesture. This allows the human operator to record the surgical gesture as (s)he is simultaneously operating the robot, reducing the time and effort to look at videos and performing the annotations. For labeling the erroneous gestures, we recorded the time that we injected the fault to one of the kinematics state variables and the time that the fault led to any of the common errors in Table II based on the video data and then mapped those times to the corresponding gestures. As a result, we were able to automate our gesture and erroneous gesture annotations for all experiments conducted in the Gazebo simulator. A total of 890 out of 4557 gestures were labeled as erroneous.



(a) Video Frames

(b) HSV Threshold of the block

(c) Contour Detection

(d) Comparison between traces, adopted from [19]

Fig. 7: Failure detection using contour segmentation and DTW

C. Metrics

We evaluated the individual components as well as the entire pipeline of our safety monitoring system in terms of **accuracy** and **timeliness** in identifying gestures and detecting errors using the metrics that are described next.

Individual Components: We trained individual components of the pipeline, namely the gesture classification and the erroneous gesture detection, separately. For the first part of the pipeline, our evaluation metrics were classification accuracy, for assessing model performance across different gesture classes, and jitter value, for identifying the timeliness of the classification. Jitter is calculated as the difference between the time our model detects a gesture and its actual occurrence, with *positive* values indicating *early detection*.

Our evaluations of the second part of the pipeline were based on the standard metrics used for binary classification: True Positive Rate (TPR), True Negative Rate (TNR), Positive Predictive Value (PPV), and Negative Predictive Value (NPV), and the Area Under the ROC Curve (AUC) of the anomaly class. We reported the micro-averages for all the metrics unless stated otherwise.

Overall Pipeline: For evaluating the classification performance of the overall pipeline, we used the F1-score as well as the AUC of the negative class. In our case, it is imperative to not classify any erroneous gestures as non-erroneous (to not miss any anomalies), while keeping the False Positive Rate (FPR) low. The F1-score, which is the harmonic mean of precision and recall, is a good indicator of how the model performs in detecting or not missing erroneous gestures. At the same time, it only reports the performance of the model using one particular threshold. As F1-score is a point-based metric, we also used AUC of ROC curves, which reports the performance over different classification thresholds.

Our metrics for assessing the timeliness of error detection were average computation time for the classifiers and *reaction time*, defined as the time to react on the advent of an erroneous gesture and calculated as the difference between the actual time of error occurrence and the time it is detected:

$$reaction_t = actual_t - detected_t \quad (4)$$

The reaction time can be used as a measure of the time budget that we have for taking any corrective actions to prevent potential safety-critical events. A *positive* value means that our model can predict an error *before* its occurrence (early detection) whereas a negative value indicates the detection of error after it has already happened (detection delay). As shown in Case 1 in Figure 8, our classifier predicts every kinematics sample as erroneous or non-erroneous. So there might be cases where different parts within the same gesture are classified as erroneous or non-erroneous. The reaction time is calculated based on the first time an erroneous sample is detected within a gesture.

We also report the percentage of times that the erroneous gestures were detected before their actual occurrence (% Early Detection in Table VIII). To calculate this, we divided the total number of times when the reaction time was positive by the total number of erroneous gesture occurrences.

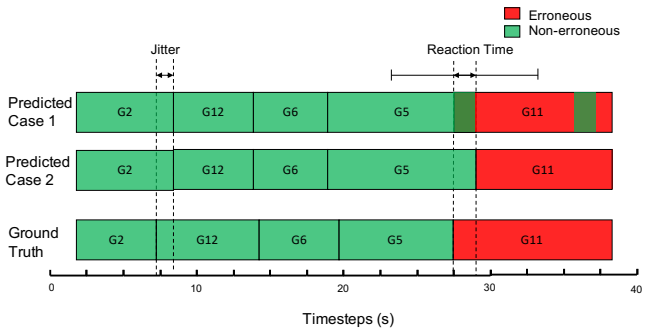


Fig. 8: Example Timeline for Detecting Anomalies

V. Results

A. Performance of Pipeline Components

Gesture Segmentation and Classification: All our results are averaged across the 5 trials of LOSO setup. Table IV shows the accuracy of our best performing model for all the tasks in the JIGSAWS dataset compared to two state-of-the-art supervised learning models that only rely on kinematics data, [44] and [45]. In addition, we also evaluated our model for the Block Transfer task on the Raven II. Our best performing model was a 2 layer stacked LSTM, with input time-step of 1, comprising of 512 and 96 LSTM units respectively, followed by a fully-connected layer with 64 units and a final softmax layer. For the tasks in the JIGSAWS dataset, the input to the model were all the 38 kinematics features from the robot manipulators. For the Block Transfer task on Raven II, we used the same LSTM architecture but the input to our model was the Cartesian Positions and Grasper Angles for each of the manipulators. [44] used a variation of the Skip-Chain Conditional Random Fields (SC-CRF) that can better capture transitions between gestures over longer periods of frames. [45] introduced Shared Discriminative Sparse Dictionary Learning (SDSDL) that aims to jointly learn a common dictionary for all gestures in an unsupervised manner together with the parameters of a multi-class linear support vector machine (SVM). For Suturing, our gesture classifier achieved competitive average accuracy of 84.49% on the test data. For Block Transfer, which has more training data and is a simpler task with no recurrence of gestures, our model achieved an accuracy of 95.16%.

Table IX shows that for the Suturing task our model detected the gestures within a jitter value of 337 ms, performing best for G2, G3, G4, and G6 with over 80% accuracy and worst for G10. Our model was unable to detect G10 which is "Loosening more suture" as it does not occur frequently (see Fig. 3a), with only 1% of transition probability from G6 and 13% transition probability from G4. In addition, as seen in Table II, there were no common errors in G10.

Erroneous Gesture Detection: We trained our erroneous gesture detection system on individual gestures, assuming perfect gesture boundaries. This allowed us to independently evaluate the performance of this module and evaluate different architectures and models that are suited for time-series classification, including LSTM networks and 1D CNNs. We also experimented with different supervised learning architectures, from kernel-based models such as SVM to ensemble techniques such as Random Forest, but here only report results for LSTM networks and 1D-CNNs for their superior performance over other architectures. We further experimented with different subsets of kinematics features, while using the set of all the features as our baseline. Our experiments specifically involved using different combinations of Cartesian Position (C), Rotation matrix (R), Grasper Angle (G) and Joint Angle (J) variables.

Method	Suturing	Knot Tying	Needle Passing	Block Transfer
This work	84.49 %	81.69 %	69.34 %	95.16 %
SC-CRF [44]	85.24 %	80.64 %	77.47 %	N/A
SDSDL [45]	**86.32 %**	82.54 %	74.88 %	N/A
Training size	102,698	44,512	66,914	4,197,988
Number of Trajectories	39	28	36	115

TABLE IV: Gesture classification accuracy in LOSO setup

Setup	Model	Layers	Features	Lr [a]	TPR	TNR	PPV	NPV
gesture specific	LSTM	512,128, 64,16*	All	1e-4	0.75	0.72	0.67	0.80
gesture specific	LSTM	128,32, 16,16*	C,R,G	1e-4	**0.76**	0.72	0.67	**0.81**
gesture specific	Conv	512,128, 32,16*	C,R,G	1e-4	0.76	0.73	0.68	0.80
gesture specific	Conv	512,128, 32,16*	All	1e-4	**0.76**	**0.73**	**0.69**	0.80
non-gesture specific	LSTM	512,128, 64,16*	All	1e-4	0.73	0.71	0.66	0.77

TABLE V: Overall performance of the erroneous gesture classification step for Suturing on the dVRK using different setups, for input time-window=5, stride=1

[a] Initial Learning Rate, * Fully-Connected Layer

Tables V and VI show the best performing models for each setup for Suturing and Block Transfer tasks, respectively. We overall observed that being gesture specific led to better accuracy (higher TPR, TNR, PPV, NPV), even with smaller datasets and that 1D-CNNs performed better than LSTM networks for binary classification of gestures for both the tasks. Training the models using specific features (Cartesian, Rotation and Grasper Angle) led to similar or better performance compared to training with all the features. In both cases, the best performing model had higher TPR and TNR while achieving competitive NPV and PPV. This suggests that the models can identify the unsafe gestures with good accuracy while not providing too many false alerts.

Table VII shows the average AUCs achieved for each gesture class using the best performing 1D-CNN model. Our model performed best on gestures G6 and G4 for Suturing. The common error for both was when the "Robot end-effector is out of sight", which occurred frequently in the demonstrations and often among surgeons with less

Setup	Model	Layers	Features	Lr [a]	TPR	TNR	PPV	NPV
gesture specific	Conv	256,128, 64,16*	C,G	1e-4	**0.62**	**0.87**	**0.65**	0.86
gesture specific	LSTM	64,32, 64,16*	C,G	1e-4	0.62	0.85	0.57	**0.89**
non-gesture specific	Conv	256,128, 64,16*	C,G	1e-4	0.59	0.85	0.58	0.85

TABLE VI: Overall performance of the erroneous gesture classification step for Block Transfer on the Raven II using different setups, for input time-window=10, stride=1

[a] Initial Learning Rate, * Fully-Connected Layer

Gesture	Train Size	% Errors	Test Size	% Errors	AUC
G1	1432	29	358	28	0.60
G2	13728	25	3432	24	0.50
G3	34921	41	8731	40	0.70
G4	13339	77	2601	79	**0.93**
G5	2717	5	680	4	0.61
G6	18923	74	4731	74	**0.93**
G8	8413	45	2104	29	0.81
G9	1769	59	443	56	0.61
G5	681976	24	151038	19	0.72
G6	394077	25	88748	21	**0.75**
G11	241067	53	53969	41	0.66

TABLE VII: Performance of the erroneous gesture classifiers

expertise. For Block Transfer, G6 had the highest accuracy, although the common gesture-specific error in this case is "Unintentional needle/object drop". We also measured the average reaction time for detecting erroneous gestures for each gesture, as shown in Table IX. In our setup, the best value would be 0, which is when the detection of erroneous gesture coincides with the start of the gesture, due to the design of our pipeline where we first detect the gesture and then the gesture-specific anomaly, if any. Since we had erroneous and non-erroneous gestures, we also calculated the average jitter for erroneous gestures, to see their difference when compared to the overall average jitter and their effect on the reaction time. For Suturing, our model performed best for gesture G4, with an average reaction time of -0.01 frames (- 0.34 ms), as well as competitive average jitter for erroneous gestures of -84 ms, followed by G1 which had an average reaction time of -6.0 frames (-167 ms). Gestures G10 and G11 had no common errors and hence no reaction times. When looking across all gestures, we noticed our model performed best for gestures which are commonly occurring in the Suturing and Block Transfer tasks and also have higher number of errors. Improvement over less common gestures, with sparse errors will be the focus of future work.

B. Overall Performance of Safety Monitoring Pipeline

We evaluated the overall performance of our Safety Monitoring pipeline for two different setups of gesture-specific and non-gesture-specific. Although we used offline data for our analysis, our system can perform the classification in real-time.

Non-Context-Specific Safety Monitoring: As a baseline, we trained a classifier with no explicit notion of context in terms of training labels, by feeding it only the kinematics data and the corresponding safe/unsafe labels. Due to the the ability of LSTM networks to recognize varying spatio-temporal patterns coupled with larger data sizes compared to gesture-specific classifiers, the classifier demonstrated some generalization and attained competitive performance (see Table VIII). An average F1-score of 0.72 and AUC of 0.71, reaction time of +6.62 frames or 221 ms, and computation time of 1.9 ms was achieved for Suturing. For Block Transfer, the classifier achieved an average F1-score and AUC of 0.73 and 0.74, respectively. The reaction time was -15.2 frames or -457 ms.

Context-Specific Safety Monitoring: In this setup, the input kinematics samples were first passed to the gesture classifier step. Having detected the gestures, their corresponding kinematics samples were sent to a separate gesture-specific classifier to identify their safety properties. As seen in Table VIII, for Suturing the average F1-score and AUC were 0.76 and 0.81, respectively, which is an improvement over the results obtained with non-context-specific setup. The average reaction time was -1.7 frames (-57 ms) and average computation time was 2.1 ms. For Block Transfer, the trend in accuracy is similar, with the gesture-specific setup achieving a higher average F1-score of 0.88 versus 0.73 and a higher AUC of 0.86 versus 0.74. The higher accuracy, as reflected by F1-score and AUC, provides more evidence (in addition to distribution analysis in Section III) to support our hypothesis about the context-specificity of errors.

The gesture-specific models had comparatively worse reaction and computation times than the non-gesture specific pipeline due to the latency introduced for identifying the context before detecting gesture-specific anomalies. Figure 8 (Case 2) provides examples of how a negative jitter associated with the detection of the gesture can result in negative reaction times. However, for Block Transfer, we were only late by -50.8 frames or 1693 ms and for Suturing, by 220 ms, while having high accuracy.

Figure 9 compares the worst, best and median performance of the context-specific and non-context-specific setups across different demonstrations, with the context-specific pipeline having an overall better performance. To get an empirical upper bound for the overall performance of the pipeline, we evaluated our entire pipeline assuming perfect gesture boundaries. As shown in Table VIII, when using perfect gesture boundaries, the average AUC improved from 0.81 to 0.83 and reaction time improved from -57 ms to 53 ms.

VI. Discussion

Our results provide encouraging evidence for the possibility of accurate and timely detection and possible preemption of erroneous gestures. Our experiments provided us with a number of key insights:

Being context-specific results in more accurate detection of erroneous gestures but worse reaction times. Table (VIII) shows that there is an improvement of 14.1% and 16.2% of AUC over non-context specific detection, for

Setup	Avg. AUC	Avg. F1	Avg. React Time (ms)	Early Detection (%)	Avg. Compute Time (ms)
Gesture-specific with perfect gesture boundaries for Suturing	**0.83** ±.14	**0.79** ±0.13	+53 ±797	38.89 %	N/A
Gesture-specific with gesture classifier for Suturing	0.81 ±.14	0.76 ±(0.13)	-57 ±1030	**43.75 %**	2.1
Non-gesture-specific classifier for Suturing	0.71 ±.16	0.72 ±0.12	**+221** ±1047	34.53 %	1.9
Gesture-specific with gesture classifier for Block Transfer	**0.86** ±.15	**0.88** ±.14	-1693 ±5670	28.26 %	3.2
Non-gesture-specific classifier for Block Transfer	0.74 ±.18	0.73 ±0.17	**-457** ±4520	**38.78** %	1.5

TABLE VIII: Evaluation of the overall pipeline with ground-truth vs. predicted gestures, compared to a non-gesture-specific approach

Gesture	Perfect Boundaries		Gesture Specific Pipeline				
	Reaction Time (ms)	F1 score erroneous gestures	Average Jitter (ms)	Gesture Detection Accuracy	Average Jitter for erroneous gestures (ms)	Reaction Time (ms)	F1 score for erroneous gestures
G1	-2050	0.69	147	45.5	-2317	-167	0.63
G2	-189	0.36	-110	81.1	-95	-703	0.33
G3	-1810	0.54	180	**90.4**	-370	-2574	0.45
G4	**0**	**0.94**	-154	86.7	-84	**-0.34**	**0.90**
G5	0	0	-130	71.2	**0**	-2367	0.09
G6	-146	0.94	-124	87.1	-136	-235	0.90
G8	-970	0.66	-337	67.4	-1842	-610	0.60
G9	-377	0.76	**46**	63.8	-474	-767	0.60
G10	N/A	N/A	N/A	N/A	N/A	N/A	N/A
G11	N/A	N/A	297	76.8	N/A	N/A	N/A
G2	N/A	N/A	397	**96.2**	N/A	N/A	N/A
G5	-708	0.75	440	95.1	-228	-2127	0.58
G6	-1562	0.80	38	96.1	-137	-2283	0.70
G11	**-307**	**0.94**	-620	80.1	**-85**	**-667**	**0.73**
G12	N/A	N/A	**6**	92.6	N/A	N/A	N/A

TABLE IX: Effect of the pipeline components on the accuracy

Suturing and Block Transfer tasks, respectively. Having the notion of context reduces the search-space for erroneous gestures, hence allowing our models to have better TPR/TNR. However, we note that finding the best trade-off between TPR/FPR from the ROC while considering the implications for surgical safety is non-trivial and requires data from real surgeries, including adverse events and close feedback from surgeons. On the other hand, the gesture-specific models result in negative reaction times (later detection of anomalies) and higher computation times due to the latency introduced for identifying the context. However, the average reaction times are still within the 1-1.5 seconds time frame. Thus, there are still opportunities for issuing timely alerts or corrective actions (when the error periods are longer) and for the acceleration of context inference stage and improving the reaction times.

Error detection with no notion of context achieves competitive performance. With an AUC of 0.71 and 0.74 for Suturing and Block Transfer tasks, the non-gesture-specific models can be considered as a good baseline. However, their high accuracy is partly due to using larger training size data (samples from all the gesture classes) and learning from similar error patterns in some of the gesture classes.

Gesture classification performance does not proportionally impact the overall error detection performance. In other words, some errors can still be detected even if the gestures are mis-classified. This is because some gestures have very similar error patterns or common failure modes.

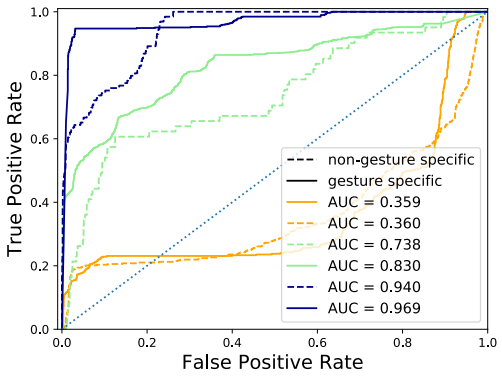


Fig. 9: Best, median and worst ROC curves for the whole pipeline in non-context-specific (baseline) and context-specific setups

For example, for Suturing, the gestures G4 and G6 both have the same failure mode where "the needle holder is not in view at all times".

Having perfect gesture boundaries leads to improved AUC and reaction time. When we look at the effect of the gesture classifier on the erroneous gesture detection (Table IX), we see that for all the gestures, having perfect classification boundaries would have resulted in better reaction times and F1 scores for erroneous gestures. This suggests possible scope of improvement in the direction of gesture classification, while also suggesting that possibly predicting the gesture boundary ahead of time could result in better reaction time. For Suturing in particular, the gestures with the highest F1 score for the gesture specific pipeline were G4 and G6, which also had high gesture detection accuracy.

Higher F1 score for detecting erroneous gestures has the highest impact on the reaction time. As seen in Table IX, misclassifying gestures or negative jitter values have less impact on the reaction time. On the other hand, the best reaction time is -0.34 ms for G4, which also had the highest F1 score for detecting erroneous gestures.

1-D CNN performs better than LSTM models for detecting erroneous gestures. Firstly, we are only classifying kinematics samples within a gesture to safe or unsafe, instead of across the entire trajectory, meaning that there is no long/short-term dependency over the class. Secondly, 1D-CNNs benefit from the feature extraction of the Convolutional layers to learn a good mapping between the gesture-specific patterns and the binary labels. Combining Convolutional layers with LSTM units would greatly increase the computational cost of the pipeline and potentially the timeliness of the monitor, thus, it was not considered here.

VII. Related Work

Safety and Security in Medical Robotics: Safety is widely recognized as a crucial system property in medical robotics. Previous work [46] introduced a conceptual framework that can capture both the design-time and run-time characteristics of safety features of medical robotic systems in a systematic and structured manner. In [7], a systems-theoretic hazard analysis technique (STPA) was used

to identify the potential safety hazard scenarios and their contributing causes in the RAVEN II surgical platform and the corresponding real adverse events reported in robotic surgery [3]. [6] proposed an anomaly detection technique based on real-time simulation of surgical robot dynamic behavior and preemptive detection of safety hazards such as abrupt jumps of end-effectors. [19] presented a monitoring system for real-time identification of subtasks using unsupervised techniques and detecting errors based on subtask-specific safety constraints learned from fault-free demonstrations. [47] presented a system to predict unsafe manipulation in robot-assisted retinal surgery by measuring small scleral forces and predicting force safety status.

Coble et al. proposed using remote software attestation for verification of potentially compromised surgical robot control software in unattended environments such as the battlefield [48]. Other works have focused on improving the security of surgical robots by introducing new networking protocols such as Secure and Statistically Reliable UDP (SSR-UDP) [49] and Secure ITP [50] that aim at increasing reliability and confidentiality of surgeon's commands.

Our work has the similar goal of early detection and mitigation of safety-critical events as [6], [19], but it is the first attempt at using supervised deep learning methods for *online* and *gesture-specific* safety monitoring. It can be used in conjunction with the mechanisms proposed by the previous works to improve resilience of surgical robots against both errors and attacks.

Surgical Workflow Analysis: Automatic analysis of surgical workflow for surgeon skill evaluation and surgical outcome prediction has been the subject of many previous works. In [21], authors modeled minimally invasive procedures as stochastic processes using Markov chains and used kinematics data along with dynamics of surgical tools for decomposing complex surgical tasks. [51] presented a feature collection, processing and classification pipeline for automatic detection and segmentation of surgical gestures (surgemes) in dry-lab settings. [52] showed that Recurrent Neural Networks can be used for the task of gesture recognition, while maintaining smooth boundaries over time. In [25], authors proposed RP-Net, a modified version of InceptionV3 model [53], for automatic surgical activity recognition during robot-assisted radical prostatectomy (RARP) procedures. [54] combined formal knowledge, represented by an ontology, and experience-based knowledge, represented by training samples, to recognize current phase of a surgery for context-aware information filtering. In this work, we focus on modeling the surgical context similar to the Markov chain models presented in [21] and on identifying the surgical gestures based on time-series data similar to [52]. However, our main goal is to detect the *erroneous gestures*.

Context-Aware Monitoring: Context-aware anomaly detection has been the focus of many recent works on safety-critical systems. For example, in [55] a context-aware reasoning framework with sensor data fusion and anomaly detection mechanisms was developed to support personalized healthcare services at home for the elderly. [56] showed that using the notion of context and incorporating usual behavior of services leads to improved detection accuracy over traditional detection mechanisms for critical service oriented architectures. [57] provided a framework for context-based detection of network intrusions by incorporating protocol context and byte sequences. Our work shares similarities with the aforementioned by incorporating context for improving anomaly detection, but it relies on deep learning for *real-time context-inference and anomaly detection* in robotic surgery.

VIII. Threats to Validity

Our solution relies on the accuracy and generalizability of DNNs for detecting the operational context followed by the context-specific errors. While DNNs have been widely successful across many domains, slight perturbations in the input data brought about by the noise in the environment [58] or attacks [59] can lead them to misclassify with high confidence. However, most of the proposed adversarial examples on DNNs target image-based classification systems. Our safety monitoring system is based on kinematics samples and we only use computer vision for orthogonal labeling of failures. A further robustness analysis and design of our ML-based safety monitor against accidental and malicious perturbations is the subject of future work.

In addition, the performance of supervised learning models heavily depends on the accurate labeling of the operational context, or surgical gestures, and the context-specific anomalies, or erroneous gestures. Our labeling of the erroneous gestures for the JIGSAWS dataset was based on the human annotations of the corresponding videos. We labeled any gesture that had an occurrence of an anomaly as erroneous even if the error did not occur at the beginning of the gesture. Future work will focus on automated labeling of trajectory data from real surgical tasks (similar to our automated labeling of Block Transfer task on RAVEN II robot using video data) for more precise localization of errors.

IX. Conclusion

We presented an end-to-end safety monitoring system for real-time context-aware identification of erroneous gestures in robotic surgery. Our preliminary results show the promise of our kinematics-only based solution in timely and accurate detection of unsafe events, even when the vision data might not be available or be sub-optimal. Our experimental results validate the need for context-aware monitoring, while also suggesting that some surgical gestures have similar error-patterns and can potentially be better monitored together as a sequence. Our results also show the potential for early detection and prevention of these unsafe events, which could be further enhanced by having access to larger training datasets and extending the semantics of context using vision or other sensing modalities. Future work will focus on the generalization of our solution to a wider set of realistic surgical gestures and tasks with a larger number of trials. We also plan to further improve the accuracy and timeliness of our safety monitoring system to enable successful prevention of safety-critical events during surgery.

REFERENCES

[1] Intuitive Surgical, Inc., "2017 Annual Report," http://www.annualreports.com/HostedData/AnnualReportArchive/i/NASDAQ_ISRG_2017.pdf.

[2] G.-Z. Yang, J. Cambias, K. Cleary, E. Daimler, J. Drake, P. E. Dupont, N. Hata, P. Kazanzides, S. Martel, R. V. Patel *et al.*, "Medical robotics—regulatory, ethical, and legal considerations for increasing levels of autonomy," *Sci. Robot*, vol. 2, no. 4, p. 8638, 2017.

[3] H. Alemzadeh, J. Raman, N. Leveson, Z. Kalbarczyk, and R. K. Iyer, "Adverse events in robotic surgery: a retrospective study of 14 years of fda data," *PloS one*, vol. 11, no. 4, p. e0151470, 2016.

[4] E. Rajih, C. Tholomier, B. Cormier, V. Samouëlian, T. Warkus, M. Liberman, H. Widmer, J.-B. Lattouf, A. M. Alenizi, M. Meskawi *et al.*, "Error reporting from the da vinci surgical system in robotic surgery: A canadian multispecialty experience at a single academic centre," *Canadian Urological Association Journal*, vol. 11, no. 5, p. E197, 2017.

[5] T. Bonaci, J. Herron, T. Yusuf, J. Yan, T. Kohno, and H. J. Chizeck, "To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots," *arXiv preprint arXiv:1504.04339*, 2015.

[6] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, and R. K. Iyer, "Targeted attacks on teleoperated surgical robots: dynamic model-based detection and mitigation," in *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016, pp. 395–406.

[7] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, J. Raman, N. Leveson, and R. Iyer, "Systems-theoretic safety assessment of robotic telesurgical systems," in *International conference on computer safety, reliability, and security*. Springer, 2014, pp. 213–227.

[8] T. R. Eubanks, R. H. Clements, D. Pohl, N. Williams, D. C. Schaad, S. Horgan, and C. Pellegrini, "An objective scoring system for laparoscopic cholecystectomy," *Journal of the American College of Surgeons*, vol. 189, no. 6, pp. 566–574, 1999.

[9] O. Elhage, B. Challacombe, A. Shortland, and P. Dasgupta, "An assessment of the physical impact of complex surgical tasks on surgeon errors and discomfort: a comparison between robot-assisted, laparoscopic and open approaches," *BJU international*, vol. 115, no. 2, pp. 274–281, 2015.

[10] P. Joice, G. Hanna, and A. Cuschieri, "Errors enacted during endoscopic surgery - a human reliability analysis," *Applied ergonomics*, vol. 29, no. 6, pp. 409–414, 1998.

[11] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011.

[12] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White, "Raven-ii: an open platform for surgical robotics research," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 4, pp. 954–959, 2012.

[13] R. P. Goldberg, M. Hanuschik, H. Hazebrouck, P. Millman, D. Kapoor, J. Zabinski, D. Robinson, D. Weir, and S. J. Brogna, "Ergonomic surgeon control console in robotic surgical systems," Feb. 21 2012, uS Patent 8,120,301.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[15] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[16] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci® surgical system," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 6434–6439.

[17] E. M. Bonrath, N. J. Dedy, B. Zevin, and T. P. Grantcharov, "Defining technical errors in laparoscopic surgery: a systematic review," *Surgical endoscopy*, vol. 27, no. 8, pp. 2678–2691, 2013.

[18] E. Bonrath, B. Zevin, N. Dedy, and T. Grantcharov, "Error rating tool to identify and analyse technical errors and events in laparoscopic surgery," *British Journal of Surgery*, vol. 100, no. 8, pp. 1080–1088, 2013.

[19] M. S. Yasar, D. Evans, and H. Alemzadeh, "Context-aware monitoring in robotic surgery," in *2019 International Symposium on Medical Robotics (ISMR)*. IEEE, 2019, pp. 1–7.

[20] D. Neumuth, F. Loebe, H. Herre, and T. Neumuth, "Modeling surgical processes: A four-level translational approach," *Artificial intelligence in medicine*, vol. 51, no. 3, pp. 147–161, 2011.

[21] J. Rosen, J. D. Brown, L. Chang, M. N. Sinanan, and B. Hannaford, "Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model," *IEEE Transactions on Biomedical engineering*, vol. 53, no. 3, pp. 399–413, 2006.

[22] J. H. Peters, G. M. Fried, L. L. Swanstrom, N. J. Soper, L. F. Sillin, B. Schirmer, K. Hoffman, S. F. Committee *et al.*, "Development and validation of a comprehensive program of education and assessment of the basic fundamentals of laparoscopic surgery," *Surgery*, vol. 135, no. 1, pp. 21–27, 2004.

[23] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, D. D. Yuh *et al.*, "Jhu-isi gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling," in *MICCAI Workshop: M2CAI*, vol. 3, 2014, p. 3.

[24] C. E. Reiley and G. D. Hager, "Decomposition of robotic surgical tasks: an analysis of subtasks and their correlation to skill," in *M2CAI workshop. MICCAI, London*, 2009.

[25] A. Zia, A. Hung, I. Essa, and A. Jarc, "Surgical activity recognition in robot-assisted radical prostatectomy using deep learning," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 273–280.

[26] A. J. Hung, J. Chen, Z. Che, T. Nilanon, A. Jarc, M. Titus, P. J. Oh, I. S. Gill, and Y. Liu, "Utilizing machine learning and automated performance metrics to evaluate robot-assisted radical prostatectomy performance and predict outcomes," *Journal of endourology*, vol. 32, no. 5, pp. 438–444, 2018.

[27] A. Zia, L. Guo, L. Zhou, I. Essa, and A. Jarc, "Novel evaluation of surgical activity recognition models using task-based efficiency metrics," *International journal of computer assisted radiology and surgery*, pp. 1–9, 2019.

[28] N. Ahmidi, P. Poddar, J. D. Jones, S. S. Vedula, L. Ishii, G. D. Hager, and M. Ishii, "Automated objective surgical skill assessment in the operating room from unstructured tool motion in septoplasty," *International journal of computer assisted radiology and surgery*, vol. 10, no. 6, pp. 981–991, 2015.

[29] M. J. Fard, S. Ameri, R. Darin Ellis, R. B. Chinnam, A. K. Pandya, and M. D. Klein, "Automated robot-assisted surgical skill evaluation: Predictive analytics approach," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 14, no. 1, p. e1850, 2018.

[30] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Evaluating surgical skills from kinematic data using convolutional neural networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 214–221.

[31] K. Moorthy, Y. Munz, A. Dosis, F. Bello, A. Chang, and A. Darzi, "Bimodal assessment of laparoscopic suturing skills," *Surgical Endoscopy And Other Interventional Techniques*, vol. 18, no. 11, pp. 1608–1612, 2004.

[32] M. R. Kwaan, D. M. Studdert, M. J. Zinner, and A. A. Gawande, "Incidence, patterns, and prevention of wrong-site surgery," *Archives of surgery*, vol. 141, no. 4, pp. 353–358, 2006.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[34] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," in *Robotics Research*. Springer, 2018, pp. 91–110.

[35] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in neural information processing systems*, 2013, pp. 190–198.

[38] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] F. Chollet *et al.*, "Keras," 2015.

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[44] C. Lea, G. D. Hager, and R. Vidal, "An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks," in *2015 IEEE winter conference on applications of computer vision*. IEEE, 2015, pp. 1123–1129.

[45] S. Sefati, N. J. Cowan, and R. Vidal, "Learning shared, discriminative dictionaries for surgical gesture segmentation and classification," in *MICCAI Workshop: M2CAI*, vol. 4, 2015.

[46] M. Y. Jung, R. H. Taylor, and P. Kazanzides, "Safety design view: A conceptual framework for systematic understanding of safety features of medical robot systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1883–1888.

[47] C. He, N. Patel, I. Iordachita, and M. Kobilarov, "Enabling technology for safe robot-assisted retinal surgery: Early warning for unsafe scleral force," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3889–3894.

[48] K. Coble, W. Wang, B. Chu, and Z. Li, "Secure software attestation for military telesurgical robot systems," in *2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*. IEEE, 2010, pp. 965–970.

[49] M. E. Tozal, Y. Wang, E. Al-Shaer, K. Sarac, B. Thuraisingham, and B.-T. Chu, "On secure and resilient telesurgery communications over unreliable networks," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2011, pp. 714–719.

[50] G. S. Lee and B. Thuraisingham, "Cyberphysical systems security applied to telesurgical robotics," *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 225–229, 2012.

[51] H. C. Lin, I. Shafran, D. Yuh, and G. D. Hager, "Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions," *Computer Aided Surgery*, vol. 11, no. 5, pp. 220–230, 2006.

[52] R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G. D. Hager, "Recognizing surgical activities with recurrent neural networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2016, pp. 551–558.

[53] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[54] D. Katić, A.-L. Wekerle, J. Görtler, P. Spengler, S. Bodenstedt, S. Röhl, S. Suwelack, H. G. Kenngott, M. Wagner, B. P. Müller-Stich *et al.*, "Context-aware augmented reality in laparoscopic surgery," *Computerized Medical Imaging and Graphics*, vol. 37, no. 2, pp. 174–182, 2013.

[55] B. Yuan and J. Herbert, "Context-aware hybrid reasoning framework for pervasive healthcare," *Personal and ubiquitous computing*, vol. 18, no. 4, pp. 865–881, 2014.

[56] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Context-awareness to improve anomaly detection in dynamic service oriented architectures," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2016, pp. 145–158.

[57] P. Duessel, C. Gehl, U. Flegel, S. Dietrich, and M. Meier, "Detecting zero-day attacks using context-aware anomaly detection at the application-layer," *International Journal of Information Security*, vol. 16, no. 5, pp. 475–490, 2017.

[58] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.

[59] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.