Automated Surgical Procedure Assistance Framework Using Deep Learning and Formal Runtime Monitoring

Gaurav Gupta(✉), Saumya Shankar(✉), and Srinivas Pinisetty(✉)

Indian Institute of Technology Bhubaneswar, Bhubaneswar, India
{gg13,ss117,spinisetty}@iitbbs.ac.in

Abstract. There have been tremendous developments in minimally invasive approaches for various surgical treatments due to the benefits for patients such as less pain and faster recovery. However, surgeons face a number of obstacles while performing these surgeries, including inadequate depth perception, limited range of motion, and difficulty gauging the force to be delivered in the tissue. As a result, improved support for these surgeries is needed to provide surgeons with automated assistance, reducing complications and needless patient damage.

In this work, we propose an approach, leveraging deep learning and formal methods, to develop an automated surgical procedure assistance framework. To the best of our knowledge, our framework is the first to develop an automated surgical procedure assistant using deep learning and formal methods. We use Faster R-CNN to identify the surgical instruments/tools used to perform the surgical procedure. Based on the high-level description of the crucial guidelines that should be obeyed during a good surgical procedure, we obtain the monitoring code that identifies a bad behaviour in a surgical procedure using formal monitor synthesis techniques. For example, any violation in the tools' usage during the surgical procedure can alert the surgeons to take immediate corrective measures. To illustrate the practical applicability of the proposed approach, we consider the case of cholecystectomy (laparoscopic) surgery and illustrate how our framework can assist a surgeon during a laparoscopic surgical procedure. We implemented the proposed framework, and validated its technical feasibility using (offline) video samples of the surgical procedure from the modified Cholec80 dataset.

Keywords: Deep learning · Faster R-CNN · Formal methods · Laparoscopic surgery · Surgical tool detection

1 Introduction

In the surgical field, nowadays people not only care about the treatment results, but also the comfort and minimal invasion during the treatment, which has given

This work has been partially supported by IIT Bhubaneswar Seed Grant (SP093).

T. Dang and V. Stolz (Eds.): RV 2022, LNCS 13498, pp. 25–44, 2022.
https://doi.org/10.1007/978-3-031-17196-3_2

rise to the new era of Minimally Invasive Surgeries (MISs). MIS emerged in the 1980s, which include surgical techniques that limit the size of incisions needed. Here, surgeons make tiny cuts in the skin, and insert small tools, cameras, and lights to operate the patient. In the last 20 years, many surgeons have come to prefer MIS over the traditional (open) surgeries since, it requires smaller incisions and cause less pain, scarring, and damage to healthy tissue. Thus, the patient will have a faster recovery and shorter hospital stays (up to 50% reductions for some procedures [17]).

There have been constant innovations to improve MIS that include technological innovations in instruments used (such as laparoscopic instruments), novel clinical measurements, and MIS-associated technologies (such as surgical robotics, image guidance systems, and advanced signal processing methods). However, MIS is accompanied with many visualization and control challenges (images from the camera are from unnatural positions with unintuitive scale). There are problems like inadequate depth perception, limited range of motion, and difficulty gauging the force to be delivered in the tissue. These are often time taking and have risks of complications due to anesthesia, bleeding and infections.

MIS require surgeons to have a particular skill set to gain excellence and optimized outcomes. Experience and assistance (manual or automated) during surgeries reduces operative time and complications, especially in complex surgeries. Thus, giving high confidence to the surgeons. The assistance provided during surgeries mainly includes, experts giving immediate feedback on the ongoing surgery performed by surgeons. According to a study report[1], there is a critical shortage of expert surgeons (as supervisors), and unavailability of considerable time from them to observe and provide feedback on the surgeries performed by the trainee surgeons. Thus, there is a need to adopt the automated way of providing feedback to the surgeons during the surgery.

The automated assistance through observation and feedback, may include various Artificial Intelligence (AI) techniques to analyze a surgical procedure. For example, Machine Learning (ML) models can be trained on surgical procedure videos, which can detect various patterns and forecast health risks/illness as well as treatments, to ease the overall process for surgeons. In the last decade, we have seen tremendous improvements in the field of AI/ML. These ML techniques have been applied in robot-assisted surgeries [22] for better performances. For example, ML techniques have successfully performed detection of the surgical tools, so that we can analyse the movement of each tool during a complex surgery and can generate feedback for the surgeons [23]. This kind of rich analysis using Deep Learning (DL) [19] can help upgrade the surgical procedure. These ML approaches combined with techniques providing rigorous correctness guarantees, such as formal methods, help in building robust and reliable systems.

Designing and developing critical systems require the use of formal methods and model-driven developments. Since formal specification languages have a

[1] The Complexities of Physician Supply and Demand: Projections From 2018 to 2033, Prepared for the AAMC by IHS Markit Ltd., June 2020, https://www.aamc.org/media/45976/download?attachment.

precise syntax and semantics, formally defining policies will make them clearer. It will eliminate ambiguities and inconsistencies, and make them more easily amenable to automatic processing and code synthesis with certain correctness guarantees.

Although there are few frameworks [3,38], with absolute guarantees of correctness, but these do not provide specific automated feedback to the surgeons during a surgery. Thus, we propose a framework for providing automated surgical procedure assistance, using DL and formal runtime monitoring approaches. We use Faster R-CNN [41], a region-based Convolutional Neural Network (CNN) to identify/detect the surgical instruments/tools that appear during a surgery. The critical policies that should be followed in a good surgical procedure are formally defined (expressed as Valued Discrete Timed Automata, see Sect. 4). Based on the monitor synthesis approaches proposed in [29,37], the runtime monitors are generated directly from these policies. These monitors will take the array of identified tools from CNN and detect any violation of the critical policies during the surgical procedure and alert the surgeons about it. This ensures inappropriate behaviour (deviations from policies) during a surgical procedure are identified.

To demonstrate the adaptable use of the proposed approach, we consider the case of cholecystectomy (laparoscopic) surgeries. Cholecystectomy is a type of laparoscopic surgery performed in the gall bladder, with the aid of a laparoscope (video camera). We illustrate how our framework can assist a surgeon during a laparoscopic surgical procedure, by identifying bad behaviours and alerting the surgeons about it. We implemented the proposed framework[2], and the technical feasibility of the approach is validated using (offline) video samples of the surgical procedure from the modified Cholec80 dataset [46].

2 Overview of the Proposed System

As discussed in Sect. 1, we work on automated surgical procedure assistance framework in a (real-time) surgery using DL and formal runtime monitoring approaches. We do so by providing feedback to the surgeons during a surgery, so that it can be carried out effectively even without the expert's supervision. Here is an overview of our work: we employ DL approaches to detect tools that are used during a surgical procedure. Based on the knowledge of the clinical guidelines from the domain expert surgeons, the key policies to be followed for safe surgical procedure can be understood. From that understanding, we formally specify the policies from which a monitor is synthesized. It will identify any bad behaviour during a surgical procedure by looking at the sequence, time of tools' occurance, etc. and intimate the surgeons about it. In this section, we give the architecture (shown in Fig. 1) of the work which comprises of two modules: tool identification module and monitoring module. We describe each module and discuss this framework for a laparoscopic surgery.

[2] The framework is available at *https://doi.org/10.5281/zenodo.6899355*.

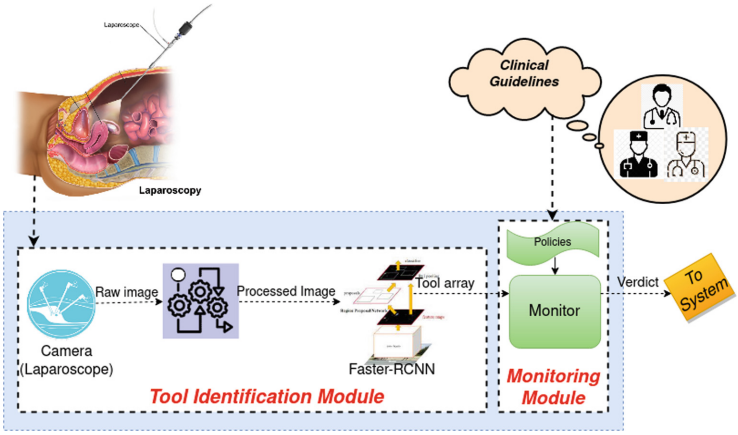

Fig. 1. Architecture of the proposed system

Software Architecture: The first module is the tool identification module. It consists of an input source (video camera) which captures frames/images in real time and processes it (processing may involve various transformations, such as image enhancement, scaling, etc.) to make it suitable to be fed to the Deep Neural Network (DNN) model. The DNN model is trained on the surgery dataset which takes the processed frame and identifies/detects the surgical instruments used in the frame (if any). It returns an array having a boolean entry for each tool's presence or absence. This array is forwarded to the monitoring module which will analyze the tool's occurrence against some policies.

The clinical guidelines for a safe surgical procedure, from domain expert surgeons can be formally defined as policies. In the monitoring module, the monitor is synthesized directly from these policies, which will take the tool occurrence array and will keep track of the specified policies being obeyed by the received arrays. It will report violations of the policies, if any. At last, the violation message is sent to the system which will convey it to the surgeon to take the corrective action.

The above architecture generalizes well to many surgeries having an annotated dataset to train the model. In order to use the proposed architecture to a specific surgery, the appropriate transformations have to be applied to the captured frames. Then, these frames have to be fed to the DNN model that has been trained on the corresponding surgery's dataset. Also, relevant policies have to be understood from the experts and specified formally, which covers "good" practices during a surgery (or resp. "bad" practices that may happen in a surgery) to synthesize a monitor.

As stated in Sect. 1, we consider laparoscopic surgeries, which are MIS, performed in the abdomen (gall bladder), with the aid of a laparoscope. A laparoscope is a lean slender shaped tool with tiny video camera and light on the tip. With few millimeters small incisions, the surgeon inserts different instruments, including the laparoscope through the abdominal wall, and performs the surgery

while visualizing it on a video screen. One can even generate a diverse multitude of realistically looking synthetic images, by image-to-image translation method, from a simple laparoscopy simulation, to gather ample data [30]. We take modified Cholec80 dataset [46] of cholecystectomy (laparoscopic) surgery, for training the DNN model of the tool identification module. For illustrating our approach, we consider the following set of example policies for the laparoscopic surgeries (specifying a good surgical behaviour):

Example 1 (Example policies). Consider simple policies P_1 and P_2, defined below, which analyzes tool usage patterns and are used to validate a surgical procedure:

- P_1: "Tool T_2 (e.g. Irrigator) should not be used after tool T_3 (e.g. Specimen Bag)";
- P_2: "Tool T_1 (e.g. Bipolar) should not be used for more than 20 t.u. continuously".

One can also include other policies, for example, a policy which keeps a count on the usage of tool in the complete surgery, etc.

In the following sections, we will see further details of each of the modules presented in the architecture (shown in Fig. 1).

3 Surgical Tool Detection

One of the most complex challenges in computer vision is object detection. Researchers have been developing various algorithms for improvements and have achieved remarkable results. Traditional object detection methods like the Viola-Jones detector (2001) [47], HOG (2006) [8] and DPM (2008) [12] have performed quite well. These methods have their limitations in dealing with images and videos because of their complex features.

Since the last decade, we have seen tremendous improvements in ML and DL techniques. When DL is used for object detection, it has shown to achieve state-of-the-art results. DL-based object detector solves object detection in two steps: 1) finds an arbitrary number of objects in the frame, 2) classify every object and estimate its size with a bounding box. One can divide these tasks into stages (like Fast-RCNN [15]) to make the process easier to achieve better results. Some methods, like YOLO [40], combine both tasks into one step to achieve better performance but at the cost of accuracy. We have used Faster-RCNN for tool detection, a two-stage object detection algorithm which gave high precision and accuracy.

3.1 Faster-RCNN

Faster R-CNN Fig. 2 is a region-based CNN technique mainly developed for object detection. The approach is inclined towards real-time object detection. Faster R-CNN introduced Region Proposal Network (RPN), which increased its

performance over Fast R-CNN [15]. RPN takes an image as input and gives a set of proposed regions where objects can be found, as output. We are using a pre-trained CNN model of VGG-16 with 13 sharable layers. It is connected to two fully connected layers- a box regression layer (reg) and a box classification layer (cls). We are using a 3×3 spatial window size on this input image to generate region proposals.

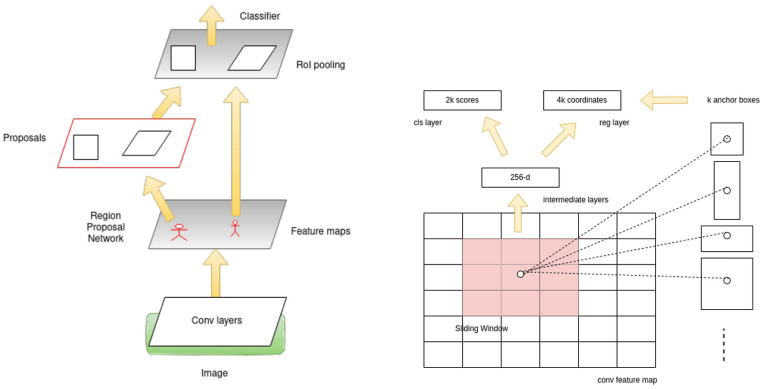

Fig. 2. Architecture of Faster R-CNN [41]

Faster R-CNN also uses the concept of anchors Fig. 2, which are fixed-sized boxes generated, having their centre at the sliding window. Anchors uses scales and aspects ratios; we have used three scales and three aspect ratios in our model, resulting in a total of 9 anchor boxes at each sliding window position. Faster R-CNN assigns a binary objectness label to each anchor, indicating the presence of an object in it. To the anchors with positive objectness label, it further assigns positive or negative labels depending on their Intersection over Union (IoU) with any ground truth box. Positive labels are assigned to anchors with IoU greater than 0.7, and negative labels are assigned to anchors with IoU less than 0.3. Only positive and negative labelled anchors are used for the training of RPN.

Now, we define our multi-task (classification and regression) loss function [41] as:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}} \sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}\left(t_i, t_i^*\right)$$

where:

- i: index of the anchor
- p_i: predicted probability of anchor i being an object
- $p_i{}^*$: label for ground truth anchor {1: positive , 0: negative}
- t_i: vector for 4 parametrised coordinates giving the position of the positive anchor

– t_i^*: ground truth box coordinates associated with a positive anchor.
– N_{cls}: normalising factor for classification loss.
– N_{reg}: normalising factor for regression loss.
– λ: balancing factor
– L_{cls}: log loss over two classes (object or not)
– L_{reg} : robust loss (smooth L1 loss)

We also parametrise each coordinate of bounding box regression as:

$$t_x = \frac{(x - x_a)}{w_a}; t_y = \frac{(y - y_a)}{h_a}$$
$$t_w = \log\left(\frac{w}{w_a}\right); t_h = \log\left(\frac{h}{h_a}\right)$$
$$t_x^* = \frac{(x^* - x_a)}{w_a}; t_y^* = \frac{(y^* - y_a)}{h_a}$$
$$t_w^* = \log\left(\frac{w^*}{w_a}\right); t_h^* = \log\left(\frac{h^*}{h_a}\right)$$

where:

– x, y are box centre coordinates
– w, h are width and height of the box
– x, x^*, x_a are for predicted box, anchor box and ground truth box respectively.

Note: We are calculating regression loss only for positively labelled anchors, that's why $p_i^* L_{reg}$.

3.2 Dataset

In 2016, M2CAI[3] had launched two open online challenges- M2CAI tool presence detection challenge and M2CAI workflow (phase) detection challenge. Cholec80 provided a dataset having 80 videos with phase and tool annotations for these challenges. Fifteen videos were used in the M2CAI tool presence detection challenge (10 for training and validation, 5 for testing). These released videos are 30 to 70 min long, having 25 frames per second (fps) but later sampled to 1 fps to obtain around 23000 total frames. These frames had been binary classified for presence or absence of 7 tools - Grasper, Hook, Scissors, Specimen bag, Bipolar, Clipper and Irrigator (shown in the Fig. 3).

We have used a modified version of this dataset, released by Stanford University [19], since we want to do both tool detection and localisation in a frame. It contains 2811 frames having spatial bounding boxes annotations around the tools done by experts. Out of these 2811 frames, 2248 and 563 are used for training and validation respectively. These frames contain at most three tools being used simultaneously. Table 1 shows number of annotated images for each tool in this dataset.

[3] Workshop and challenges on modeling and monitoring of computer assisted interventions, http://camma.u-strasbg.fr/m2cai2016/index.php/program-challenge/.

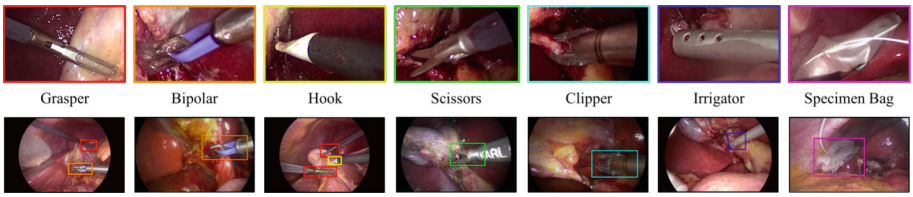

Fig. 3. Showing all 7 tools used in our dataset. (Source [19])

Table 1. Annotated images for each tool in our dataset

Tools	Number of annotated images
Bipolar	450
Clipper	400
Grasper	1422
Hook	308
Irrigator	485
Scissors	388
Specimen Bag	476

3.3 Implementation

RPNs are trained by backpropagation using Stochastic Gradient Descent (SGD). In this strategy, we use $N(= 256)$ randomly sampled positive and negative anchors. This mini-batch is selected so that the ratio of positive to negative anchors is close to 1:1, ensuring the model to be unbiased. We use transfer learning for initializing weights in our RPN. Since we use VGG-16, we initialise our first 13 layers with the weights of the model pre-trained on ImageNet classification, through which it learns to detect basic features. We fine-tune our network by training it with our dataset. In the training part, we set our learning rate to 0.001 and use ReduceLROnPlateau (built-in function of Keras) to decrease it by a factor of 10, if it remains constant for five epochs. Thus, we made our model's learning rate adaptable which improved precision and accuracy. We implemented this Faster R-CNN in our surgical tool detection. Since, the size of the image in our dataset is 334×596, thus we keep our image resizing to 450 to ensure that our trained model also works well for tool detection in surgical videos with a frame size of 460×680.

Anchors have different scales and aspect ratios. We have used anchors with box areas of 128^2, 256^2 and 512^2 pixels and with aspect ratios of 1:1, 1:2 and 2:1. These anchor boxes per sliding window were found sufficient to cover most objects.

The usage of Non-Maximal Suppression (NMS), which helps in predicting the correct box around the tools was required. We have used NMS on the proposal regions based on classification scores. Since tool localisation does not involve a

complete tool, we have set the NMS to 0.05. This means that if there are two predicted boxes for the same tool with more than 5% common area, NMS will remove one with lesser probability.

3.4 Model Performance

We evaluate our model performance on test images and videos (of different lengths varying from 30 to 70 min). We used 563 annotated surgical images and 5 videos to test our model (testing was done by comparing the boolean vector indicating the presence or absence of tools for each frame, with the given ground truth boolean vectors). Figure 4 shows 1 to 3 tools detected in various frames by our framework.

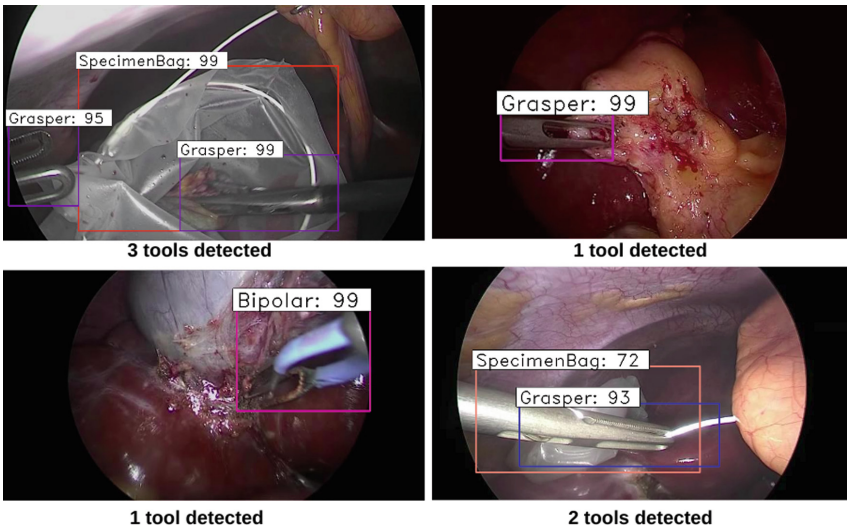

Fig. 4. Surgical tools in frames having 1 to 3 tools detected by our framework

Our model processed around 5 fps and provided an average precision and accuracy of 88.21% and 95.82% for video level detection (see Table 2).

4 Formal Runtime Monitoring

Runtime Verification (RV) [2,10] techniques allow one to check if a run of a system under observation complies with (or violates) a specified policy/property. Because the focus is on verifying the current execution/trace of the system being monitored, a formal model of the system is not required (system being monitored is usually considered as a black-box). As a result, RV techniques are lightweight, and issues like state explosion are avoided because always one (current) execution

Table 2. Model's performance on test videos

Videos	Accuracy	Precision
Video 1	96.11	90.56
Video 2	94.93	82.84
Video 3	97.60	92.73
Video 4	95.77	86.14
Video 5	94.67	88.79
Average	95.82	88.21

of the system is monitored/verified. An RV monitor does not change the system's execution/behaviour; instead, it monitors and examines whether the system's actual execution is meeting the specified properties.

Runtime Enforcement (RE) [31,32] approaches are an extension of the RV approachs, concentrating on ensuring that the executions of systems being monitored are consistent with some desired policy. An enforcement monitor converts an (untrustworthy) input execution (series of events) into a policy-compliant output sequence of events (e.g., defining a desired safety requirement). In order to do so, an enforcement monitor performs certain evasive actions to prevent the violation. These evasive actions might include blocking the execution, modifying input sequence by suppressing and/or inserting actions, and buffering input actions until a future time when it could be forwarded.

The different monitoring frameworks differ on the power of the enforcement mechanism (i.e. the different evasive actions it can take) and the supported policy specification language. For example, monitors in [11,33] allowed buffering of the input events and used automata to specify the policies ([33] used timed automata [1] to specify real-time policies); whereas monitors for reactive and cyber-physical systems in [36] allowed altering the input events and used Valued Discrete Timed Automata (VDTA) to specify the policies. VDTA supports valued signals, internal variables, and complex guard conditions, ensuring compatibility with real-world cyber-physical and industrial systems.

In this work, we employ formal runtime monitoring approaches for "assessing" a surgical procedure and thus providing assistance to the surgeon, in the form of feedback during the surgeries. We write policies and synthesize "verification" monitor out of it. We use approaches proposed in [29,37] to synthesize the monitor, as these approaches synthesize the monitoring code directly from the specified policies. The generated monitor also ensures correctness and safe behaviour in safety-critical systems. These approaches use VDTA to specify policies which is an automaton with a finite set of locations, a finite set of discrete clocks used to represent time evolution, and external input-output channels which represents system data. They also have internal variables that are used for internal computations.

We mainly use the monitor synthesis mechanism to realize some of the requirements or a component, in the overall system automatically. As illustrated

in Fig. 5, (verification and enforcement) monitors are seen as modules outside a (black-box) system, which take as input a stream of events (output of the system being monitored) and verify or correct this stream according to the policy. In this work, we mainly utilize the monitor synthesis approaches to realize a component (the monitoring module of the proposed system as shown in Fig. 5) that deals with providing feedback to the surgeons during a surgery. Instead of implementing that component, we rely on synthesis of the component from high-level policies using formal monitor synthesis approaches, so that the component is correct-by-construction.

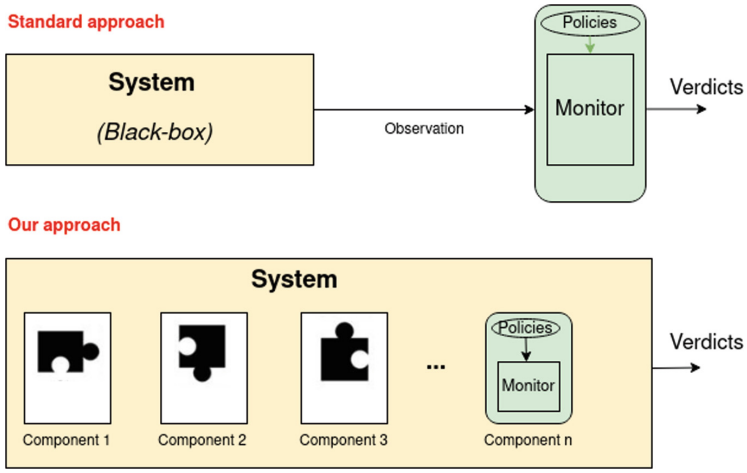

Fig. 5. Formal RV monitoring context: usual Vs. our system

Example 2. Let us define policies P_1 and P_2 of Example 1, via VDTA. Let $I = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$ be the external boolean input channels and the set of clock variables $V = \{v\}$[4] The alphabet $\Sigma = 2^I = \{0000000, 0000001, \cdots, 1111111\}$, where each event will be denoted as a bit-vector. For example, $\{T_1\} \subseteq I$ is denoted as $1000000 \in \Sigma$, and $\{T_1, T_4\} \subseteq I$ is denoted as $1001000 \in \Sigma$. Figure 6 shows policies P_1 and P_2 where, $L = \{l_0, l_1, l_2\}$ is the set of locations, with l_0 as the initial location and $\{l_0, l_1\}$ as the accepting locations in both the automata. According to policy P_2, from initial state l_0, upon input event $!T_1$[5] (indicating absence of tool T_1 in the captured frame), the system remains in same state (self-loop on l_0). The transition from location l_0 to

[4] VDTA can handle more expressive policies. Its entire potential (e.g., the output channels, the internal variables) has not been realised here.

[5] $!T_1$ denotes set of all the events in Σ, where tool T_1 is absent (bit corresponding to T_1 is 0 and other bits can be 0/1), e.g., $(0000000), (00000001), \cdots, (0111111)$. Similarly, T_1 denotes set of all the events in Σ, where tool T_1 is present (bit corresponding to T_1 is 1), e.g., $(1000000), (1000001), \cdots, (1111111)$.

l_1 is taken upon input event T_1 (indicating presence of tool T_1 in the captured frame), with clock v reset to 0. From location l_1, if the input event is $!T_1$, then the system again goes back to location l_0, otherwise remains at l_1 till the value of the clock v is < 20. With input event T_1, when v is ≥ 20, it goes to dead location[6] l_2, thus ensuring adherence to the policy. Policy P_1 can be similarly expressed as a VDTA as illustrated in Fig. 6.

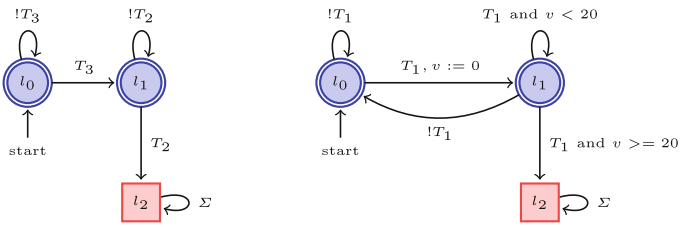

Fig. 6. Automaton for policy P_1 (on left) and P_2 (on right) (Color figure online)

Remark 1. Note that, in this work, we slightly modify the approach of monitor synthesis, proposed in [29,37].[7].

RV Monitor from VDTA: In this work we synthesize an RV monitor from policies expressed as VDTA. An RV monitor deals with checking the observation of the current execution (denoted as σ) satisfies or violates the policies and gives verdict accordingly. There are different RV frameworks, such as [2], where the verdicts provided are {*True, False, ?*}. Verdict *True* means every continuation of σ satisfies the policy, verdict *False* means there is no continuation of σ which satisfies the policy and verdict "*?*" means unknown (no conclusive verdict can be provided). There are other frameworks, such as [34], which refines the unknown verdicts into *Currently true* and *Currently false*. Verdict *Currently true* indicates that the current observation of the execution satisfies the policy but not every continuation of it satisfies the policy and similarly, *Currently false* indicates that the current observation of the execution violates the policy but not every continuation of it violates the policy.

In this work, we define and implement our RV monitoring framework for VDTAs, where a monitor for a policy defined as a VDTA takes a stream of events over Σ as input (current observation), and emits a verdict in {*True, False, Currently true, Currently false*}.

[6] A dead location (denoted by red squares throughout the paper) is a location in the automaton, from where there is no path in the automaton from that location, to reach an accepting location.

[7] We synthesize a verification monitor instead of an enforcement monitor. Thus, our model will give verdicts in the form of feedback when it detects a violation and will not exercise its power of correcting faulty inputs i.e., editing erroneous inputs/outputs using edit functions as proposed in the followed paper.

5 Experimentation

To evaluate the performance of our proposed framework, we implemented the architecture, given in Fig. 1. This section shows the experimentation of the implemented architecture. We build the module for tool identication (following Sect. 3.1), take the dataset (discussed in Sect. 3.2) and evaluate the framework against some simple policies (given in Example 1).

The captured and processed frame is supplied to the tool identification module (built using the approaches discussed in Sect. 3), which identifies the tools present in the current frame. It yields an array of boolean variables showing presence or absence of each tool. The obtained array is then passed to the monitoring module (built using the approaches discussed in Sect. 4). It contains monitors synthesized from the specified policies. For experimentation, we consider policies P_1 and P_2 defined in Example 1 of Sect. 2 with $I = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$ (the boolean tool array) and the set of clock variables $V = \{v\}$. These policies are defined in the intended format as illustrated in 1.1 and as automata illustrated in Fig. 6. The policies are combined using standard product construction [37] as $P_1 \cap P_2$ which corresponds to the conjunction of both policies P_1 and P_2. The monitor is directly synthesized for the policy defining $P_1 \cap P_2$.

The monitor (C code synthesized from high-level policies) integrated with the system takes the tool array and checks if the policies are obeyed or not. It will raise an alert if the surgical procedure is not carried out according to the specified policies.

```
interface of tool_detection {
  in bool T1, T2, T3, T4, T5, T6, T7;
  out int16_t res; }


policy p1 of tool_detection {
  states {
    s0 {
      -> s0 on !T3;
      -> s1 on T3;}
    s1 {
      -> s1 on !T2;
      -> violation on (T2) recover res := 1;}}}

policy p2 of tool_detection {
  internals {
    dtimer_t v;}
  states {
    s0 {
      -> s1 on T1 : v:=0;
      -> s0 on !T1;}
    s1 {
      -> s1 on (T1 and v<20);
```

```
        -> s0 on !T1;
        -> violation on (T1 and v>=20) recover res := 2;}}}
```

Listing 1.1. Policies in the intended format

For example, consider policy P_1 and received input trace $\sigma = (0000110) \cdot (0010010) \cdot (0100000)$. From initial location l_0, upon input event "0000110", the automaton remains in l_0; from l_0, upon input event "0010010", the automaton makes a transition to l_1, (the bit corresponding to tool T_3 is 1, indicating presence of tool T_3); and from l_1, upon input event "0100000", the automaton goes to dead location l_2, (the bit corresponding to tool T_2 is 1, indicating presence of tool T_2), since it violated the policy (Policy P_1 abstained the use of a tool T_2 after tool T_3, which may signify carelessness that should be avoided). Thus, in this case the monitor will give verdict as *False* and will reset itself and continue with the next event.

Upon reception of a *False* verdict, the system will raise an alert that the surgery is not being performed as per the specified policies.

Similarly, when a surgeon uses tool T_1 for a longer (than usual) period of time (which may point to rough handling of tissue resulting in tissue damage), then it will violate policy P_2 and the monitor will raise a similar alert.

These alerts will give instant feedback to the surgeon, and he can be more careful during the surgical procedure; thus providing assistance to the surgeon. One can set different types of alerts depending on the type of policies. For example, for some critical policies, where violating the policies can have severe consequences, alerts can be sending notifications to the senior surgeons; whereas, for the other soft policies, the alerts can be allocating resources (likely to be used in future) to avoid preoperative delays.

Performance Discussion: Our proposed system spends considerable time in tool identification only, since it employs DL, whose execution time is more dependent on the processing power of the available architecture. Our architecture/machine (Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz with Quadro RTX 4000 and 8GB Graphics RAM/GPU) can process around 5 fps. Thus, the overall architecture ensures real-time feedback to the surgeons with the considered policies.

Moreover, our proposed architecture can utilize state-of-the-art trained model benefits and further enhance its overall performance. In one recent published work on surgical tool detection using cascading of CNN [48], researchers were able to detect tools in around 0.023 s. Use of this model in our tool identification module will further bring down frame processing time. Thus, it can process more than 40 fps.

The framework is implemented and is available for download at *https://doi.org/10.5281/zenodo.6899355*.

6 Related Work

Several object detection and formal runtime monitoring approaches are related to the one used in this paper. We give a comparison with approaches for these in Sect. 6.1 and Sect. 6.2 respectively.

6.1 Object Detection

Object detection has received a lot of research attention in recent years because of its tight association with the well known video analysis and image processing techniques. Object detection tasks can be broken down into two stages - object localization (location of objects in the image) and object classification (category of each object present).

Handcrafted features and shallow trainable structures are the foundations of traditional object identification systems. Feature extraction in these traditional methods uses SIFT [26], HOG [8] and Haar-like [25]. Classification of objects is done using traditional classifiers like Supported Vector Machine (SVM) [7], AdaBoost [13], Deformable Part-based Model (DPM) [12] etc. These methods have their limitation e.g. it cannot obtain complex features in images or videos.

With the advent of DL [24], image classification improved significantly. Because of the deep architecture of DNN and R-CNN [16], these were able to easily learn the complex features. Also, large datasets and robust training methods got rid of the need of manual feature extraction. Many improvements in the R-CNN models have been proposed. Fast R-CNN [15] further optimizes classification and bounding box regression tasks. Faster R-CNN [41] generates region proposals using an additional subnetwork. YOLO [40] uses a fixed-grid regression to detect objects. These models have made real-time object detection feasible with improved accuracy.

Robot-assisted surgeries will be widely used in the future and surgical tool detection is the first step towards it. DL approaches have shown excellent results in improving its performance. The M2CAI tool detection challenge[8] helped to achieve new benchmarks in surgical tools detection. There are several studies done to address tool detection in videos and obtain a richer analysis of surgeries. Sarikya et al. [43] used multi-modal CNN for localization and fast detection of tools in Robot-assisted surgeries. In 2019, Zhao et al. [49] proposed a method using two CNNs designed for detecting coarse and fine locations of surgical tools. Jin et al. [19] used a two-stage framework of Faster R-CNN to localize and detect tools in a frame, which was later used for surgical skill assessment in a surgical video. Chao et al. [5] proposed a one stage framework using a modified YOLO [40] architecture, which increased the detection speed of surgical tools to 48.9 fps. Nwoye et al. [28] used CNN + Convolutional LSTM (ConvLSTM) to model temporal dependencies in the motion of surgical tools, which resulted in multiple tools tracking and detection simultaneously.

6.2 Formal Runtime Monitoring

Formal verification is the process of checking whether a design satisfies the specified requirements/policies. The policies can be expressed using high-level formalisms such as automata [29,33,35–37,44] or as temporal logic [2].

[8] Tool presence detection challenge results, Workshop and Challenges on Modeling and Monitoring of Computer Assisted Interventions, http://camma.u-strasbg.fr/m2cai2016/index.php/tool-presence-detection-challenge-results/.

Model checking [6] is an automated static formal verification approach to check if the policies are satisfied by an abstract formal model of the system or not. RV [2,10] and RE [31,32] approaches do not require a formal model of the system since only a single execution of the system is considered. RV (RE) approaches checks at runtime if a run of a system under scrutiny satisfies a given correctness policy or not (enforces them in the latter case).

DL is increasingly used in domains like autonomous driving [18], healthcare [14], cybersecurity [27], etc. Designing these learning based systems that have strong, provable, assurances of correctness w.r.t. the policies is always a focal point. This has increased the discussions on verified artificial intelligence [42,45]. The very first attempt to formally verify a neural network was done by Pulina and Tacchella [39]. They came up with techniques to verify that the output of a fully-connected neural network with sigmoid activations is always within specified safety bounds. Later, [9,20,21], proposed approach for the verification of neural networks employing specific activation functions.

The adoption of DL in medical and health-care systems to facilitate a procedure has increased complexity of the system and made it more susceptible to errors. Use of formal methods, along with advanced design, control and deployment paradigms, is often recommended to guarantee the correctness and safety of these medical systems. In the attempt, Bresolin et al. [3] discussed the applications of formal methods to verify the properties of control systems built for autonomous robotic systems that performed surgeries. They demonstrated the automatic execution of simple tasks like puncturing. Brunese et al. [4] represented patient magnetic resonances as formal models and predicted the prostate cancer Gleason score. Pore et al. [38] proposed a safe deep reinforcement learning framework that guarantees safety criteria for automated surgical tasks.

The works in [3,38] deals with autonomous robot-assisted surgeries safeguarded by formal methods. But performing fully automated surgeries is quite complex and less reliable. Upskilling of the existing surgeons by providing assistance to them during complex surgeries can be really helpful. To the best of our knowledge, our framework is the first to develop an automated surgical procedure assistant using DL and formal methods. Thus, reducing complications and needless patient damage during MIS.

7 Conclusion and Future Work

This paper presents a complete framework using DL and runtime monitoring for developing an automated surgical procedure assistant. In this approach, we use Faster R-CNN to identify the surgical tools used to perform the surgical procedure. Then, we specify some policies on tools' usage pattern that should be obeyed during a good surgical procedure. We obtain a monitoring code out of the policies. It will identify a bad behaviour in a surgical procedure. This way, we can catch any violation in the tool's usage during the surgical procedure and can alert surgeons to take immediate corrective measures.

To illustrate the practical applicability of the proposed framework, we consider the case of cholecystectomy (laparoscopic) surgery. We have implemented

the framework and have evaluated its technical feasibility using some example properties on (offline) video samples of the surgical procedure from the modified Cholec80 dataset.

In the future, we plan to enrich this framework with diverse policies to cover a large number of safe behaviours and then use this framework for robot assisted surgeries.

Acknowledgment. Gaurav expresses gratitude to Prof. Neelam Sinha for her invaluable guidance during the summer research internship at International Institute of Information Technology, Bangalore organised by Indian Academy of Sciences. The work done there has helped to generate new ideas for this paper.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8, https://www.sciencedirect.com/science/article/pii/0304397594900108
2. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. **20**, 14 (2011). https://doi.org/10.1145/2000799.2000800
3. Bresolin, D., Geretti, L., Muradore, R., Fiorini, P., Villa, T.: Formal verification of robotic surgery tasks by reachability analysis. Microprocess. Microsyst. **39** (2015). https://doi.org/10.1016/j.micpro.2015.10.006
4. Brunese, L., Mercaldo, F., Reginelli, A., Santone, A.: Formal methods for prostate cancer gleason score and treatment prediction using radiomic biomarkers. Magn. Resonan. Imaging **66**, 165–175 (2020). https://doi.org/10.1016/j.mri.2019.08.030, https://www.sciencedirect.com/science/article/pii/S0730725X19302437
5. Choi, B., Jo, K., Choi, S., Choi, J.: Surgical-tools detection based on convolutional neural network in laparoscopic robot-assisted surgery, vol. 2017, pp. 1756–1759 (2017). https://doi.org/10.1109/EMBC.2017.8037183
6. Clarke, E.M., Henzinger, T.A., Veith, H.: Introduction to Model Checking. In: Handbook of Model Checking, pp. 1–26. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_1
7. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1023/A:1022627411411
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR 2005, vol. 1, pp. 886–893 (2005). https://doi.org/10.1109/CVPR.2005.177
9. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 269–286. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_19
10. Falcone, Y., Fernandez, J.-C., Mounier, L.: Runtime verification of safety-progress properties. In: Bensalem, S., Peled, D.A. (eds.) RV 2009. LNCS, vol. 5779, pp. 40–59. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04694-0_4
11. Falcone, Y., Mounier, L., Fernandez, J.C., Richier, J.L.: Runtime enforcement monitors: composition, synthesis, and enforcement abilities. Formal Meth. Syst. Des. **38** (2011). https://doi.org/10.1007/s10703-011-0114-4
12. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010). https://doi.org/10.1109/TPAMI.2009.167

13. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997). https://doi.org/10.1006/jcss.1997.1504, https://www.sciencedirect.com/science/article/pii/S002200009791504X
14. Ghassemi, M., Naumann, T., Schulam, P., Beam, A., Chen, I., Ranganath, R.: A review of challenges and opportunities in machine learning for health, May 2020
15. Girshick, R.: Fast r-cnn (2015). https://doi.org/10.1109/ICCV.2015.169
16. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation, November 2013. https://doi.org/10.1109/CVPR.2014.81
17. Grunstad, J.: Two new studies reveal benefits of laparoscopic surgery for uterine cancer, March 2006
18. Huang, Y., Chen, Y.: Autonomous driving with deep learning: a survey of state-of-art technologies (2020)
19. Jin, A., et al.: Tool detection and operative skill assessment in surgical videos using region-based convolutional neural networks, March 2018. https://doi.org/10.1109/WACV.2018.00081
20. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
21. Khedr, H., Ferlez, J., Shoukry, Y.: Effective formal verification of neural networks using the geometry of linear regions. CoRR abs/2006.10864 (2020). https://arxiv.org/abs/2006.10864
22. Klodmann, J., et al.: An introduction to robotically assisted surgical systems: current developments and focus areas of research. Current Robot. Rep. **2**(3), 321–332 (2021). https://doi.org/10.1007/s43154-021-00064-3
23. Lavanchy, J., et al.: Automation of surgical skill assessment using a three-stage machine learning algorithm. Sci. Rep. **11** (2021). https://doi.org/10.1038/s41598-021-84295-6
24. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015). https://doi.org/10.1038/nature14539
25. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Proceedings of the International Conference on Image Processing, vol. 1, p. I (2002). https://doi.org/10.1109/ICIP.2002.1038171
26. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94
27. Macas, M., Wu, C.: Review: Deep learning methods for cybersecurity and intrusion detection systems (2020)
28. Nwoye, C.I., Mutter, D., Marescaux, J., Padoy, N.: Weakly supervised convolutional LSTM approach for tool tracking in laparoscopic videos. Int. J. Comput. Assist. Radiol. Surg. **14**(6), 1059–1067 (2019). https://doi.org/10.1007/s11548-019-01958-6
29. Pearce, H., Pinisetty, S., Roop, P.S., Kuo, M.M.Y., Ukil, A.: Smart I/O modules for mitigating cyber-physical attacks on industrial control systems. IEEE Trans. Industr. Inf. **16**(7), 4659–4669 (2020). https://doi.org/10.1109/TII.2019.2945520
30. Pfeiffer, M., et al.: Generating large labeled data sets for laparoscopic image processing tasks using unpaired image-to-image translation. In: Shen, D., et al. (eds.) MICCAI 2019. LNCS, vol. 11768, pp. 119–127. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32254-0_14

31. Pinisetty, S., Falcone, Y., Jéron, T., Marchand, H.: Runtime enforcement of regular timed properties. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 1279–1286. SAC 2014, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2554850.2554967
32. Pinisetty, S., Falcone, Y., Jéron, T., Marchand, H., Rollet, A., Nguena Timo, O.L.: Runtime enforcement of timed properties. In: Qadeer, S., Tasiran, S. (eds.) RV 2012. LNCS, vol. 7687, pp. 229–244. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35632-2_23
33. Pinisetty, S., Falcone, Y., Jéron, T., Marchand, H., Rollet, A., Timo, O.N.: Runtime enforcement of timed properties revisited. Formal Meth. Syst. Des. **45**(3), 381–422 (2014). https://doi.org/10.1007/s10703-014-0215-y
34. Pinisetty, S., Jéron, T., Tripakis, S., Falcone, Y., Marchand, H., Preoteasa, V.: Predictive runtime verification of timed properties. J. Syst. Softw. **132**, 353–365 (2017). https://doi.org/10.1016/j.jss.2017.06.060, https://www.sciencedirect.com/science/article/pii/S0164121217301310
35. Pinisetty, S., Preoteasa, V., Tripakis, S., Jéron, T., Falcone, Y., Marchand, H.: Predictive runtime enforcement. Formal Meth. Syst. Des. **51**(1), 154–199 (2017). https://doi.org/10.1007/s10703-017-0271-1
36. Pinisetty, S., Roop, P., Smyth, S., Tripakis, S., von Hanxleden, R.: Runtime enforcement of reactive systems using synchronous enforcers, November 2016
37. Pinisetty, S., Roop, P.S., Smyth, S., Allen, N., Tripakis, S., Hanxleden, R.V.: Runtime enforcement of cyber-physical systems. ACM Trans. Embed. Comput. Syst. **16**(5s) (2017). https://doi.org/10.1145/3126500
38. Pore, A., et al.: Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery. In: 2021 IEEE/RSJ International Conference on IROS, pp. 4025–4031 (2021). https://doi.org/10.1109/IROS51168.2021.9636175
39. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 243–257. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_24
40. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection, June 2016. https://doi.org/10.1109/CVPR.2016.91
41. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 1, pp. 91–99. NIPS 2015. MIT Press, Cambridge, MA, USA (2015)
42. Russell, S., et al.: Letter to the editor: research priorities for robust and beneficial artificial intelligence: an open letter. AI Mag. **36**, 3 (2015). https://doi.org/10.1609/aimag.v36i4.2621
43. Sarikaya, D., Corso, J., Guru, K.: Detection and localization of robotic tools in robot-assisted surgery videos using deep neural networks for region proposal and detection. IEEE Trans. Med. Imaging 1 (2017). https://doi.org/10.1109/TMI.2017.2665671
44. Schneider, F.B.: Enforceable security policies. ACM Trans. Inf. Syst. Secur. **3**(1), 30–50 (2000). https://doi.org/10.1145/353323.353382
45. Seshia, S.A., Sadigh, D., Sastry, S.S.: Towards verified artificial intelligence (2020)
46. Twinanda, A., Shehata, S., Mutter, D., Marescaux, J., De Mathelin, M., Padoy, N.: EndoNet: a deep architecture for recognition tasks on laparoscopic videos, February 2016. https://doi.org/10.1109/TMI.2016.2593957

47. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on CVPR 2001, vol. 1, p. I (2001). https://doi.org/10.1109/CVPR.2001.990517
48. Zhao, Z., Cai, T., Chang, F., Chen, X.: Real-time surgical instrument detection in robot-assisted surgery using a convolutional neural network cascade. Healthc. Technol. Lett. **6** (2019). https://doi.org/10.1049/htl.2019.0064
49. Zhao, Z., Voros, S., Chen, Z., Cheng, X.: Surgical tool tracking based on two CNNs: from coarse to fine. J. Eng. 2019 (2019). https://doi.org/10.1049/joe.2018.9401