Formal Probabilistic Analysis of a Virtual Fixture Control Algorithm for a Surgical Robot

Muhammad Saad Ayub and Osman Hasan

School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Islamabad, Pakistan
{saad.ayub,osman.hasan}@seecs.nust.edu.pk

Abstract. With the ever-growing interest in the usage of minimally-invasive surgery, surgical robots are also being extensively used in the operation theaters. Given the safety-critical nature of these surgeries, ensuring the accuracy and safety of the control algorithms of these surgical robots is an absolute requirement. However, traditionally these algorithms have been analyzed using simulations and testing methods, which provide in-complete and approximate analysis results due to their inherent sampling-based nature. We propose to use probabilistic model checking, which is a formal verification method for quantitative analysis of systems, to verify the control algorithms of surgical robots.The paper provides a formal analysis of a virtual fixture control algorithm, implemented in a neuro-surgical robot, using the PRISM model checker. We have been able to verify some probabilistic properties about the out-of-boundary problem for the given algorithm and found some new insights, which were not gained in a previous attempt of using formal methods in the same context. For validation, we have also done some experiments by running the considered algorithm on the Al-Zahrawi surgical robot.

Keywords: Probabilistic Model Checking, PRISM, Surgical Robotics.

1 Introduction

Minimal-invasive surgery (MIS) [16] is a surgical procedure in which a laparoscope (a thin lighted tube), along with a high resolution camera, and other surgical instruments are inserted into the human body through small incisions rather than a relatively larger incision commonly used in the traditional open surgeries. The internal operating field is then visualized on a video monitor connected to the scope. MIS has become quite popular these days as it facilitates quick patient recovery and has less chances of post-operative infections. However, these added benefits come at the cost of highly precise movements required by the surgeons in the confined space provided. Robotic arms and hands have a high degree of dexterity and thus are playing a promising role in facilitating surgeons for operating in very tight spaces in the body that would otherwise only be accessible through open (long incision) surgery. Operations relevant to microanatomy and neuro-endoscopy are specifically performed through robotic-assisted MIS because of the static nature of human skull. Moreover, treating the

brain tumor via small hole surgery and precise robotic arms also reduces the risk of damaging the brain tissue overlying the tumor.

Despite the extreme precision of surgical robots, these man-made machines have their own inherent inaccuracies. There is always a risk that these robotic arms may go out of control and damage other organs instead of working in the surgical area. This problem is termed as the *out-of-boundary problem*. As these robots are operated by humans via a software interface, so the substantial loss of force feedback (haptics) and a lack of adaptability are the most common risk factors that lead to the *out-of-boundary problem* [12]. These issues may lead to life threatening situations[10]. The conventional approach to test the out-of-boundary problem is by manually operating the robot. The more the user operates the robot, the more are the chances of finding errors but this method is very time consuming and it also does not ensure a reliable system behavior for all possible scenarios. The other most commonly used analysis method for finding the out-of-bound errors is computer simulation [24], where a computer-based model of the robot is tested systematically but this method is very expensive in terms of computational resources and memory, due to the continuous and randomized nature of the problem. Moreover, it is also not possible to simulate each and every case for success and failure and thus most of the times an incomplete analysis is done by leaving a significant number of test cases.

Formal verification methods [8], i.e., computer-based methods for mathematical analysis for systems, have been used to overcome the above-mentioned inaccuracy limitations for many hardware and software systems. Model checking [8] is based on state-space exploration methods and is one of the most widely used formal methods. The system under verification is mathematically modeled as state-transition system. This model is then used within a computer to automatically verify that it meets rigorous specifications of intended behavior [8].

We propose to conduct the formal analysis of control algorithms used in surgical robots using probabilistic model checking. The proposed framework allows us to capture the uncertainties of the real-world scenarios using Markovian models and verify probabilistic properties within the sound environment of a probabilistic model checker. The quantitative information provided by these probabilistic properties can play a vital role in designing safer and more performance efficient surgical robots. In particular, the paper provides the formal probabilistic verification of a control algorithm for the neuro-mate robot that is used to perform skull surgeries[24]. We verified the deadlock freedom, reachability, out-of-boundary and collision freeness properties. Moreover, we validated our results by conducting real experiments on the Al-Zahrawi robot [9].

2 Related Work

Given the safety-critical nature of robotic applications, formal verification methods have been widely used to conduct their analysis. For example, Mikaël [17] used probabilistic model checking to verify the flexibility property of swarm robots in a collective foraging scenario. Kim et al. [11] developed the discrete

control software of the Samsung's home robot (SHR) using Esterel and used the XEVE model checker to verify the stopping behavior of SHR. Webster et al. [23] verified the autonomous decision making system of a personal home robot using the SPIN model checker. Scherer et al. [22] built a method for the verification of robotic control software based on the Java path finder. They verified the safety and liveness properties for a line following robot. Model checking has also been used to verify the motion planning algorithms of various robots. Lahijanian et al. [14] verified the probability of the robot reaching its destination via a safe path. Similarly, Fainekos et al. [4] addressed the problem of generating continuous trajectories for mobile robots while satisfying formulas in temporal logic using the NuSMV model checker. Saberi et al. [21] used the mCRL2 language [6] to create a formal model for a multi-robot system by creating different communicating processes and the Modal u-calculus [5] to formally specify *deadlock freedom*, *collision-freeness* and the *reachability* properties. Li [15] used the HOL4 theorem prover to verify the collision freeness property for collision-free motion planning algorithm (CFMC) of a dual-arm robot.

In the context of surgical robotics, Bresolin et al. [3] used hybrid automata [1] to formalize an autonomous surgical robot and analyzed the surgical task of puncturing, i.e., the method of piercing a biological tissue with the help of a needle. Similarly, a formal modeling and verification approach for the virtual fixture control algorithm for a surgical robot has been reported in [12]. The authors used a hybrid logic, i.e., differential dynamic logic and quantified differential dynamic logic to model the system and verify it using the KeymaeraD [19] theorem prover. They showed that the algorithm is unsafe and modified it to satisfy safe operation. This work modeled and analyzed the real-time dynamics of the system quite well but ignored the randomized aspects, such as the input from the surgeon (force exerted and direction of motion). The main focus of the current paper is to overcome this gap and provide quantitative information about the formally verified properties of control algorithm of surgical robots.

3 Preliminaries

This section gives a general overview of probabilistic model checking and the virtual fixture based control algorithm that is formally verified in the paper.

3.1 Probabilistic Model Checking and PRISM

Probabilistic Model Checking [18] is used for the formal analysis of systems that exhibit random behavior and thus can be represented as Markov chains [13]. The probabilistic behavior of systems can be captured via discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs) and probabilistic timed automata (PTAs) [18]. Once the Markovian model of the system under verification is finalized, then the probabilistic properties of the system are formally specified. The commonly used specification language for probabilistic model checking is Probabilistic Linear

Temporal Logic (PLTL). The model and property of the system, expressed in the language of the probabilistic model checker, is then given to the model checker. The tool explores the model exhaustively to check all possible executions and the probabilistic queries are solved through numerical methods [18].

PRISM [13] is a widely used probabilistic model checker that supports DTMCs, CTMCs, MDPs and PTAs. It allows describing the probabilistic behavior of the given system using the reactive modules formalism [2]. PRISM incorporates state-of-the-art symbolic data structures and algorithms, based on Binary Decision Diagrams (BDDs) and Multi-Terminal Binary Decision Diagrams (MTBDDs), and its discrete-event based simulation engine provides support for statistical model checking. The components of the given distributed system are modeled as modules, which can either be synchronous or asynchronous in nature. Each module mainly consists of variables and commands. The variables describe the possible states that the module can be in and the commands describe its behaviour, i.e., the way in which the state changes over time. Variables in PRISM can be declared both globally and locally. PRISM supports (finite ranges of) integer or Boolean as data-types. Moreover, multiple instances of modules can also be instantiated. Verification properties are expressed in PRISM using the probabilistic computation tree logic (PCTL). Once a property is formulated, then the PRISM tool automatically verifies that the property conforms to the model or not. The verification results can also be logged and plotted [13].

3.2 Virtual Fixture Control Algorithm for Surgical Robots

Surgeries are usually conducted in a specific zone, which is identified for the surgical robot using a virtual boundary, usually known as the *virtual fixture* [20]. With the aid of these virtual fixtures, the robot manipulator is guided to move within the specified region [24]. A surgeon describes the operating volume by a series of planes oriented and positioned in space. These planar boundaries are divided into three zones [12, 24]: *Safe zone* is safe for the movement of robot. *Forbidden zone* is out-of-bound for the robot. *Slow zone* is the region between the safe and the forbidden zones where the movement is somewhat restricted.

The control algorithm exhibits different behaviors in the above-mentioned zones. In the safe zone, the controller allows the robot to move freely. In the slow zone, as the boundary of the forbidden zone approaches, the controller increases the resistance for movement while alarming the surgeon that she is getting closer to the boundary and also prevents the robot from crossing it [12]. The equation governing the control circuit in this region is as follows

$$\overline{p}\prime = K(\overline{p})G(\overline{f})\overline{f} \tag{1}$$

Where overbars indicate vectors and the prime ($\prime$) indicates a derivative with respect to time. p is the position and $p\prime$ is the velocity of the tip of the surgical tool attached to the robot. f is the force applied by the surgeon on the robot manipulator. G is the scaling factor, which controls the precision of the tool tip. The value of G should be high when the surgeon desires to have flexibility to

move rapidly and should be low when fine movements are required. K is the gain term, which is used to impose motion constraints on the tool . It is taken as an identify matrix in the safe zone and zero in the forbidden zone, respectively. Whereas in the slow zone, K is chosen such that the velocity is scaled down by a factor proportional to the distance of tool from the forbidden zone. The behavior of K can be abstracted as the following equation:

$$K = \frac{d}{D} \tag{2}$$

Where d is the distance of the tool from the forbidden zone boundary at any instant and D is the width of the slow zone region. The equation 1 works fine in preventing the tool from crossing the safety boundary but once the tool is in the slow zone, it attenuates motion in all the directions. Therefore the equation was modified so that once the tool enters the slow zone, the control algorithm restricts the movement of the tool in the direction of the forbidden region and allows free movement in the direction opposite to the forbidden region. This behavior is implemented using the following equation where $n1$ is unit normal to the boundary.

$$\overline{p}\prime = \overline{p}\prime - (1 - \frac{d}{D})(\overline{p}\prime.n1)n1 \tag{3}$$

The purpose of this paper is to verify probabilistic properties related to the above equation using probabilistic model checking.

4 Formalization of the Virtual Fixture Algorithm using the PRISM Language

The first step in modeling the given virtual fixture algorithm, explained in Section 3.3, is the translation of Equations (1) and (3). After some arithmetic simplification and decomposing the force and velocity into the Cartesian plane, we obtain the following equations:

$$px = G(\frac{dx}{Dx})fx, \quad py = G(\frac{dy}{Dy})fy, \quad pz = G(\frac{dz}{Dz})fz \tag{4}$$

The second step is to develop a model for the control algorithm. We have chosen to model the given algorithm as a DTMC. The virtual fixtures are defined using the Cartesian plane, where the origin is taken as the center point of the safe zone as the surgeon is quite likely to start from the center. Considering the Cartesian coordinates, the boundaries for each plane may lie on the positive axis or the negative axis. Thus for each plane, we defined four boundaries, i.e., two for the safe zone and the other two for the forbidden zone. The movement of the tool in the virtual fixture is modeled using a grid of blocks, which represent 1 unit of movement. The distance of the tool from the boundary and the boundaries of the safe and the forbidden zones are thus determined by the number of blocks on the grid as illustrated in Figure 1 for a 45x45 grid.

The scaling factor G in Equation 4 is responsible for translating the force applied by the surgeon to the velocity of the tool. It depends on the virtual

fixture area and the force applied by the surgeon. If the area of operation is small then the scaling factor is kept small so that a sudden force applied by the surgeon is not completely translated into tool velocity. The scaling factor changes with the amount of force or movement applied by the surgeon on the control stick. As the maximum force applied is limited by the movement of the control stick due to mechanical constraints, the scaling factor is responsible generating variable velocities for movement of the operating tool. In our model, the scaling factor is taken as a constant since the area of the robot is fixed and the force applied by the surgeon is non-deterministic and not limited by mechanical constraints. The model consists of three main components: the force

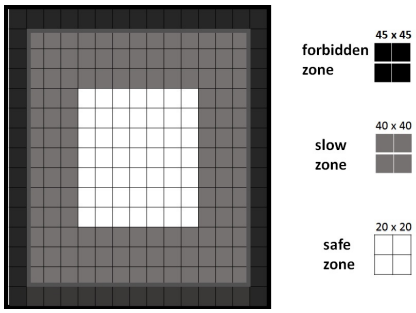


Fig. 1. Virtual Fixture Zones

module that is responsible for generating the force applied by the surgeon and the velocity translation module that converts the force applied by the surgeon into the tool velocity. It is also responsible for introducing the damping factor in the velocity. The position update block changes the current position of the tool based on the velocity and previous position and also checks the boundaries of operation. The modules are implemented as Finite State Machines (FSM) with augmented probabilities. The environment is modelled by defining the bounds of all the zones as shown in Figure 1. Data Sharing among various modules of the Control Algorithm is done via variables created in each module.

4.1 Force Module

The force module captures the behavior of the interaction of the surgeon with the system, which includes the behavior of the force applied by the surgeon's hand on the controlling tool. The force applied is further divided into three components based on the Cartesian coordinates, i.e, f_x, f_y, f_z. The force applied by the surgeon is non-deterministic with probabilistic bounds, such that the probability of the force applied by the surgeon in the direction of force applied previously is higher than the force applied in the opposite direction. Based on surgical statistics [7], we used a probability of 0.75 for the tool to retain the previous direction of movement and a probability of 0.25 for a change.The force

f=* represent non-determinism in the case when the force is zero initially.
$1 : [](f = 0) \rightarrow 1/n : f = *;$
$2 : [](f > 0) \rightarrow 0.75 : f >= 0 + 0.25 : f < 0;$
$3 : [](f < 0) \rightarrow 0.25 : f > 0 + 0.75 : f <= 0;$

4.2 Velocity Module

The velocity module determines the instantaneous velocity of the tool using the force exerted by the surgeon and the position of the tool. The velocity is also divided into three components, i.e., v_x, v_y, v_z. The control algorithm under verification is basically modeled in this module. If the position of tool is within the safe zone represented by **sl** and **sh** as the safe zone upper and lower limits in a single axis, then the force applied is directly translated to velocity. If the position of the tool is in the slow zone defined by **bl** and **bh** as the upper and lower limits for a single axis and the force is applied in the direction of the boundary, then the translated velocity is attenuated based on the scale factor K. If the tool crosses the boundary of the slow zone then the velocity is completely nullified restricting further movement in the forbidden zone.
$1 : [](p > sl \quad \& \quad p <= sh) \rightarrow (v' = gain * f);$
$2 : [](p <= sl \quad \& \quad p > bl) \quad \& \quad f <= 0 \rightarrow (v' = (dl/Dl) * gain * f));$
$3 : [](p > sh \quad \& \quad p < bh) \quad \& \quad f >= 0 \rightarrow (v' = (dh/Dxh) * gain * f));$
$4 : [](p <= sl \quad \& \quad p > bl) \quad \& \quad f > 0 \rightarrow (v' = gain * f);$
$5 : [](p > sh \quad \& \quad p < bh) \quad \& \quad f < 0 \rightarrow (v' = gain * f);$
$6 : [](p <= bl \quad \& \quad f >= 0) \rightarrow (v' = gain * f);$
$7 : [](p <= bl \quad \& \quad f < 0) \rightarrow (v' = 0);$
$8 : [](p >= bh \quad \& \quad f > 0) \rightarrow (v' = 0);$
$9 : [](p >= bh \quad \& \quad f <= 0) \rightarrow (v' = gain * f);$

4.3 Position Module

The position module determines the number of blocks on the grid that the tool will move depending on the calculated velocity. It is also divided into three components, i.e., p_x, p_y, p_z. If the position of the tool is within the limits specified by the grid size defined by **n**, then the tool is allowed to move based on the velocity. However, if it is at an edge of the grid then its movement is restricted towards the end of grid but it is allowed to move in the opposite direction freely.
$1 : [](p + v < n \quad \& \quad v > 0) \rightarrow (p' = P + v);$
$2 : [](px + v > -n \quad \& \quad v < 0) \rightarrow (p' = P + v);$
$3 : [](v = 0)|(p + v >= n)|(p + v <= (-n)) \rightarrow (p' = P);$

4.4 Multiple Surgical Tools

Most of the surgical procedures involve multiple robotic arms that are independently controlled. In order to formally model this scenario, we replicate the above-mentioned modules for force, velocity and position and allow them to run

concurrently. The tool boundary limits are considered to be the same for both tools in our model. The modules are initialized such that both tools operate simultaneously and independently; a choice that makes collision a possibility as well. We enhanced the control algorithm with collision avoidance capabilities by treating the location of one tool as a forbidden zone boundary for the other and vice versa. This will ensure that when a tool is moving towards the other tool its velocity will be attenuated to avoid collision between the two tools. The attenuation will increase as the tool gets nearer to the other tool. Thus, in essence, the main modeling concept is basically to treat the previously considered static forbidden boundaries as dynamic ones using a module `obstacle`, which creates boundary points from the other tool's position.

$1 : []ox < n \quad \& \quad ox > -n \rightarrow (ox' = ax1) \quad \& \quad (oy' = ayl);$

These boundary points are then used in the `velocity` module as additional boundaries for the model. The `velocity` module then restricts the movement of the tool if they are moving towards the other tool by a factor M, which is the ratio of distance between both tools and the maximum distance between both tools. The maximum distance is computed depending on the width of the slow zone and the distance between both tools is computed in each iteration. This will ensure that the tools are less likely to collide with each other.

$$M = \frac{d_{obs}}{D_{obs}} \tag{5}$$

Where d_{obs} is the distance between both tools with a maximum value of D_{obs}.

5 Virtual Fixture Control Algorithm

In this section, the formal model of control algorithm, described in the previous section, is verified using property specifications introduced in the proposed methodology. We verified these properties using PRISM 4.1.2 on Windows 7 64-bit operating system running on an Intel Core2 Quad Q9100 processor at 2.66 GHz with 4.0 GB of RAM. The grid size is taken as 45x45, the range of the width of the slow zone is taken to be 0 to 20 and the maximum value of the force is taken to be 6.

5.1 Deadlock Freedom

We verified the deadlock freedom of our virtual fixture control algorithm by using the built in deadlock property of the PRISM model checker. This property checks if for some states the transition from the present to future state will result in a deadlock. Our algorithm was found to be deadlock free.

5.2 Reachability

This property ensures that the surgical robot will move to the position desired by the surgeon in a finite number of steps. The presence of two robotic arms in

the virtual fixture makes the verification of the property quite important. The fact that the control algorithm, under consideration, attenuates the movement of tool, makes the verification of the reachability property very important as it may happen that the algorithm does not allow the tool to reach some areas, especially the ones that are very close to the boundaries where the attenuation is the maximum. The reachability property can be verified by checking that the tool moves from its current position and reaches the required destination in a finite number of steps if the required force is applied to it. We verified this property by associating a reward with every step of the algorithm, i.e., a reward of 1 is added to the existing reward value at every step of the algorithm. The reachability property, based on the reward accumulated along a particular path, can now be expressed as:

```
R=? [px=0 & fx>0 -> F px=(width zone limit/2)-1]
```

This property states that if the tool position is 0 and a force is applied in the positive direction, then the tool will eventually reach the boundary of the forbidden zone in a bounded number of steps or rewards. The property is not verified probabilistically as the result will not clearly depict if the tool reaches the boundary in minimal number of steps or not, while using the reward based approach we can ensure the tool will reach the boundary in a limited number of steps. The property is verified for the x-plane while observing the impact of varying the width of the slow zone. Checking this property returns the reward or number of steps that the algorithm would take to get to the edge of the forbidden zone. The properties for the y and z-planes are given as follows

```
R=? [py=0 & fy>0 -> F py=(width zone limit/2)-1]
R=? [pz=0 & fz>0 -> F pz=(width zone limit/2)-1]
```

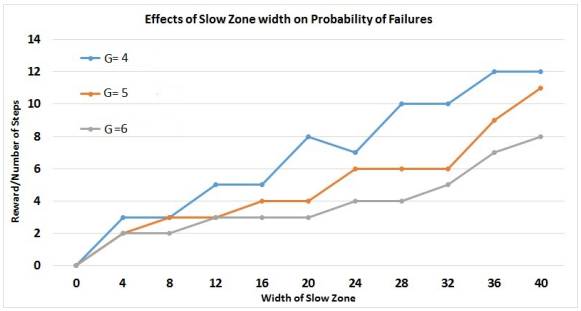


Fig. 2. Impact of Changing the Width of Slow Zone on the Reachability rewards vs width of slow zone property

These properties are verified for different widths of slow zones and the resultant rewards for the x-plane, while keeping the value of the maximum force

constant. Figure 2 shows that the rewards calculated are always a finite number and their value increases with the increase in the width of the slow zone. This is because as the width increases, the distance from origin to the edges of the virtual fixture increases and the steps to reach them also increase. This is because at each step towards the boundary of the forbidden zone, the attenuation in the velocity increases and the tool moves slowly towards the boundary, therefore, more steps are required to cater for this attenuation. These verification results show that the algorithm under verification satisfies the reachability property.

The properties are also verified by varying the scaling factor and Figure 2 shows the resultant rewards. It is observed that increasing the scaling factor increases the velocity of the tool and thus the reward value decreases. This happened because as the velocity increases the tool moves more distance in a single iteration and thus requires less number of steps to reach the destination.

5.3 Out-of-Boundary

As described previously, the main focus of this paper is the formal probabilistic analysis of the out-of-boundary problem. The most important aspect of any surgical robot is to stay within the operable area at all times. If it is allowed to move out of the operable area it may damage sensitive organs, which may lead to the loss of human life in worst-case scenarios. The given algorithm is therefore checked in the proposed methodology for boundary crossovers and their probability. In the context of our modeling, the problem can be stated as follows: At any given time during the operation, if the surgeon starts in the safe zone then the tool should not cross the boundary of forbidden zone. This property can be formally expressed in terms of the boundary limits defined for our virtual fixtures. We can simply check that the position of the tool is within these limits in every state, i.e.,

```
forall (px< bxh & px>bxl)
```

where px is the position of tool in the x-plane, bxh is the higher boundary limit and bxl is the lower boundary limit of the slow zone. The same condition should be checked for the y-plane and z-plane.

```
forall (py< byh & py>byl), forall (pz< bzh & pz>bzl)
```

The main issue with these properties is that they will either be True or False. In the case of failure, we would not know the probability of failure, which is a desirable performance characteristic as well. This limitation can be overcome by verifying the probability of failure of this property:

```
P=? (px>0 & px>sxh & fx>0 => F px>bxh)
```

Where P is the probability of failure, sxh is the boundary of the safe zone and fx is the force applied by the surgeon. This property checks the probability of crossing the boundary of the forbidden zone if the tool is in the slow zone and accelerating towards the forbidden zone. The same property can be checked for the y-plane and z-plane as follows:

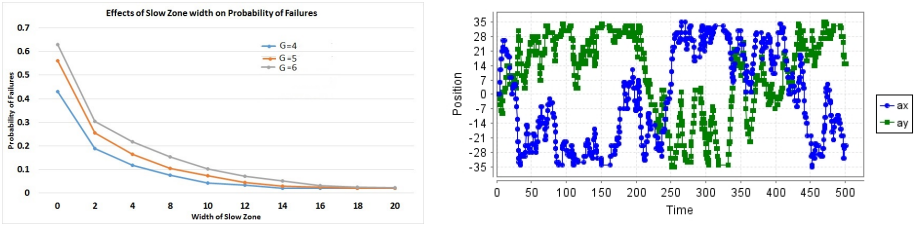
```
P=?(py>0 & py>syh & fy>0 =>F py>byh),P=?(pz>0 & pz>szh & fz>0 =>F pz>bzh)
```

These properties are for a boundary in the positive plane for each axis. The corresponding properties for the negative planes are as follows:

```
P=?(px<0 & px<sxl & fx<0 =>F px<bxl),P=?(py<0 & py<syl & fy<0 =>F py<byl)
                P=?(pz<0 & pzl<szl & fz<0 =>F pz<bzl)
```

The size of the virtual fixture and its boundaries have a great impact upon the verification time and computational requirements. Therefore, in order to avoid the state-space explosion problem, the maximum size of the virtual fixture has to be bounded and the boundaries for the safe and forbidden zones have to be varied accordingly.

Figure 3(a) shows the resultant probabilities computed after the verification of the above-mentioned properties at different slow zone widths and Scaling factor. It is seen that when the width of slow zone is increased, the probability of the surgical tool crossing the boundary decreases (Figure 3(a)). The probabilities change by varying the scaling factor. However, after a certain width of the slow zone, the probabilities become constant. This happens because as the width of the slow zone increases the chances of the surgical tool to enter the forbidden region, due to a sudden change of the velocity in the direction of the forbidden region, decreases. But no matter how much the width is increased, if the tool is at the very edge of the forbidden region and sudden changes of velocity occur in the direction of the boundary, then the tool would always cross it. Therefore, the probability does not reach zero. This is validated by simulating the extreme cases using PRISM. These probabilities are also found to be dependent on the force applied by the surgeon. This shows that the algorithm will not restrict the surgical tool from crossing the boundary if the surgeon exhibits sudden changes near the boundary of the forbidden region. The control algorithm needs to be updated to cater for these cases.



(a) **Effects of the Slow Zone Width on Probability of Failures** (b) **Position of the tool(Dx=20,Dy=17,Fmax=6,Fmin=-6)**

Fig. 3. Results of Verification of Out-of-Boundary Properties

Figure 3(b) shows the simulation of the virtual fixture model for a width of 35 units for the slow zone in the x-axis and 35 units for the slow zone in

the y-axis and a scaling factor of 6, respectively. The results show that the tool crosses the boundary of the forbidden zone, i.e., 35 units, in cases where maximum force is applied towards the boundary from the very edge, whereas the tool remains within the boundary for other cases. The results also show that the control algorithm does not ensure complete safety of the tool, i.e., it does not take into account the extreme cases, which results in the penetration of the tool in the forbidden zone. Probabilistic analysis played a vital role in identifying these extreme cases as the non-probabilistic formal techniques can only tell us if the algorithm is safe or not.

5.4 Collision Freeness

In a laparoscopic surgical operation, more than one tool are inserted inside the patient. The corresponding control algorithm is supposed to ensure that these tools do not collide with each other inside the patient. Instead of formally verifying the collision freeness property for the robotic arms, we verify the probability associated with the event when the tools collide with one another. In particular, this will compute the probability that the tools share a same grid point. This property can be defined in the context of our model by ensuring that, at any given time during the operation, the tools should not share the same position in any zone. The property can be stated by considering the position of one tool as a boundary point for other tools. For two tools, the property can be specified as

```
forall (px1!=px2)
```

where $px1$ and $px2$ are the positions of the first and second tool in the x-, respectively. The same properties are verified for the y and z-planes.

```
forall (py1!=py2), forall (pz1!=pz2)
```

These properties result in either True or False and do not give us information about frequency of failures. In order to find quantitative information in this regard, we compute the probability of failure associated with these properties

```
P=?(px1>sxl & px1<sxh & px2>sxl & px2<sxh => F px1=!px2)
```

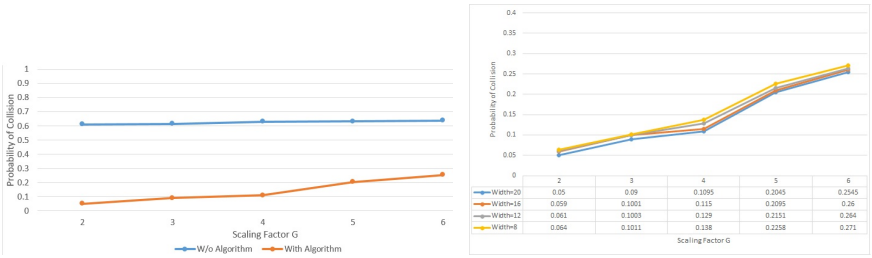
Where P is the probability of failure, $px1$ is the position of first tool, $px2$ is the position of second tool and sxh and sxl are the upper and lower limits of virtual fixture, respectively. This property is also checked for the y and z-plane.

```
P=?(py1>syl & py1<syh & py2>syl & py2<syh => F py1=!py2)
P=?(pz1>szl & pz1<szh & pz2>szl & pz2<szh => F pz1=!pz2)
```

The size of the virtual fixture and the force applied have a great impact on the verification of this property. In order to avoid the state-space explosion problem, the size of the virtual fixture is fixed and the maximum force applied is varied by changing the scaling factor. These properties are verified for both models, i.e., with and without obstacle avoidance algorithm, using different force limits keeping the width of virtual fixtures and the boundaries constant.

Figure 4(a) shows the probabilities associated with the above-mentioned properties when verified for the model with and without obstacle avoidance algorithm at different scaling factors. The probability of collision for the model without algorithm remains almost constant by varying the scaling factor as there is no restriction on collision and changing the scaling factor will not affect the collisions. On the other hand, for the case with the collision avoidance algorithm, it is observed that as the scaling factor increase the probability of collision increases (Figure 4(a)). This happens because the tools become more likely to share the same grid point when the tools are near and the velocity of one of the tool is high in the direction of the other.

Figure 4(a) clearly shows that, with the obstacle avoidance algorithm, the probability of collision decreases but does not approach zero. The properties are also verified for different widths of slow zones. The resultant probabilities are shown in Figure 4(b). It is observed that the width of slow zone does not affect the collisions of tools and the probabilities of collision are almost the same.



(a) **Effect of Force on Probability of Collision for Both Models**

(b) **Effect of Force on Probability of Collision for Different Widths of the Slow Zone**

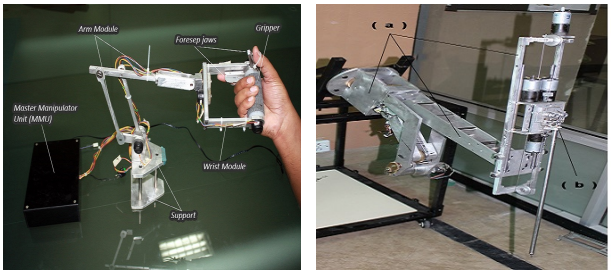
Fig. 4. Results of Verification of Collision Freeness Properties

6 Testing on Al-Zahrawi

In order to validate our verification results, we tested the considered virtual fixture control algorithm on a minimal invasive surgery (MIS) robot Al-Zahrawi [9], named after a renowned arab surgeon Abu al-Qasim Khalaf ibn al-Abbas Al-Zahrawi (936-1013), who is also known as the father of modern surgery. The Al-Zahrawi robot consists of a Master Console (MC) and Slave Console (SC) as shown in Figure 5. The master console is used to track the force applied by the surgeon and transfer it to the slave console. The surgeon operates the tool using the master manipulator and a screen to display the camera output. The master manipulator, shown in Figure 5(a), is made up of a mechanical mechanism and optical encoders to track the movements of the hand of the surgeon. It offers 6 degree of freedom, i.e., Pitch, Yaw, Roll, back/forth and individual forceps jaw

open/close. The slave console, shown in Figure 5(b), is used to reproduce the force applied by the surgeon on the patient and is mainly composed of a servo motor based mechanical structure to replicate the movements of the surgeon's hand on the patient side. The slave console provides the same degrees of freedom as the master console.

We implemented the virtual fixture based control algorithm on the Al-Zahrawi surgical robot for our experiment. The master manipulator sets the value of attenuation for the velocities based on the feedback of the positions of slave manipulators and sends them to the slave manipulator. The slave manipulator is equipped with a clamper, which is a widely used surgical instrument. Our testbed consists of five different positions, one at the center and four at the boundaries of a rectangular region. In our experiment, 40 different subjects, with various levels



(a) **Master Manipulator** (b) **Slave Manipulator**

Fig. 5. Consoles of the Al-Zahrawi Surgical Robot [9]

of expertise in robotic surgery, are asked one-by-one to pick an object, placed in the center, using the telesurgical tool and move it to any edge box and try to place it at the center of that box. The user then picks the object and places it in the box located at the opposite corner of the testbed.

The resultant boundary crossings of all the operators are logged and plotted in Figure 6 for both the cases, i.e., with the virtual fixture control algorithm and without the algorithm. The results show that the number of boundary crossings of the robotic tool without the algorithm are much greater than the ones of with the algorithm. They also show that the tool does cross the boundary with the algorithm but the crossings in that case are very less compared to ones without the algorithm. This validates our verification results, given in Section 6, stating that the algorithm is not completely safe with respect of restricting the robot within the operating area and crossovers will occur if significant force is applied near the edge of the boundary.

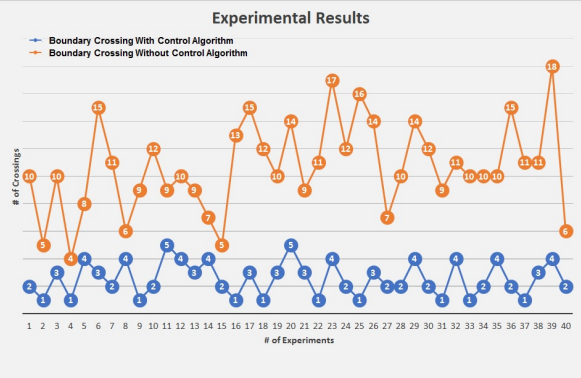


Fig. 6. Experimental Results

7 Conclusions

This paper presents a formal verification technique for a virtual fixture based control algorithm used in a surgical robot [24]. In order to consider the randomized nature of the environment, such as the force, applied by the surgeon, and its direction, we propose to use probabilistic model checking for the verification. The main idea is to first develop a formal Discrete-Time Markov chain (DTMC) model of the given algorithm and its environment. This model can then be used to analyze the corresponding probabilistic properties. The paper describes the details about modelling a well-known virtual fixture based control algorithm and also identifies the corresponding probabilistic properties. Our results confirm that the properties of out-of-boundary are failing but under certain conditions the probability of failure is very small, and thus it is quite safe to use the robot under these conditions. Since traditional model checking cannot be used to verify probabilistic properties so these insights about the safe conditions cannot be obtained.

References

1. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid Systems. pp. 209–229. Springer Berlin Heidelberg (1993)
2. Alur, R., Henzinger., T.A.: Reactive modules. Formal Methods in System Design 15(1), 7–48 (1999)
3. Bresolin, D., Guglielmo, L.D., Geretti, L., Muradore, R., Fiorini, P., Villa, T.: Open problems in verification and refinement of autonomous robotic systems. In: Euromicro Conference on Digital System Design. pp. 469–476 (2012)
4. Fainekos, G.E., Gazit, H.K., Pappas, G.J.: Temporal logic motion planning for mobile robots. In: Robotics and Automation. pp. 2020–2025 (2005)
5. Groote, J.F., Mateescu, R.: Verification of temporal properties of processes in a setting with data. In: Algebraic Methodology and Software Technology AMAST. pp. 74–90 (1999)

6. Groote, J.F., Mathijssen, A., Reniers, M., Usenko, Y., Weerdenburg, M.V.: The formal specification language mCRL2. Citeseer (2007)
7. Haidegger, T., Benyó, B., Kovács, L., Benyó, Z.: Force sensing and force control for surgical robots. In: Symposium on Modeling and Control in Biomedical Systems. pp. 401–406 (2009)
8. Hasan, O., Tahar, S.: Formal Verification Methods. Encyclopedia of Information Science and Technology, IGI Global, pages 7162–7170 (2014)
9. Hassan, T., Hameed, A., Nasir, S., Kamal, N., Hasan, O.: Al-zahrawi: A telesurgical robotic system for minimal invasive surgery. Systems Journal, IEEE,10(3) pp, 1035–1045 (2106)
10. Kazanzides, P., Zuhars, J., Mittelstadt, B., Taylor, R.H.: Force sensing and control for a surgical robot. In: Robotics and Automation. pp. 612–617 (1992)
11. Kim, M., Kang, K.C., Lee, H.: Formal verification of robot movements-a case study on home service robot shr100. In: Robotics and Automation. pp. 4739–4744 (2005)
12. Kouskoulas, Y., Renshaw, D., Platzer, A., Kazanzides, P.: Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In: Hybrid systems: computation and control. pp. 263–272 (2013)
13. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification. pp. 585–591 (2011)
14. Lahijanian, M., Wasniewski, J., Andersson, S.B., Belta, C.: Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In: Robotics and Automation. pp. 3227–3232 (2010)
15. Li, L., Shi, Z., Guan, Y., Zhao, C., Zhang, J., Wei, H.: Formal verifcation of a collision-free algorithm of dual-arm robot in hol4. In: Robotics and Automation (ICRA). pp. 1380–1385 (2014)
16. Mack, M.J.: Minimally invasive and robotic surgery. The Journal of Americal Medical Association 285(5), 568–572 (2001)
17. Mikaël, L.: Formal Verification of Flexibility in Swarm Robotics. Thesis, Department of Computer Science,Universit libre de Bruxelles (2012)
18. Oldenkamp, H.A.: Probabilistic model checking: A comparison of tools. Master's thesis, University of Twente, Enschede, Netherlands (2007)
19. Platzer, A., Quesel, J.D.: Keymaera: A hybrid theorem prover for hybrid systems (system description). In: Automated Reasoning, pp. 171–178. Springer (2008)
20. Rosenberg, L.B.: Virtual fixtures: Perceptual tools for telerobotic manipulation. In: Virtual Reality Annual International Symposium. pp. 76–82 (1993)
21. Saberi, A.K., Groote, J.F., Keshishzadeh, S.: Analysis of path planning algorithms : A formal verification-based approach. In: Robotics and Automation ICRA. pp. 232–239 (2013)
22. Scherer, S., Lerda, F., Clarke, E.M.: Model checking of robotic control systems. In: International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS). pp. 5–8 (2005)
23. Webster, M., Dixon, C., Fisher, M., Salem, M., Saunders, J., Koay, K., Dautenhahn, K.: Formal verification of an autonomous personal robotic assistant. In: Formal Verification and Modeling in Human-Machine Systems: Papers from the AAAI Spring Symposium (FVHMS 2014). pp. 74–79 (2014)
24. Xia, T., Baird, C., Jallo, G., Hayes, K., Nakajima, N., Hata, N., Kazanzides, P.: An integrated system for planning, navigation and robotic assistance for skull base surgery. ournal of Medical Robotics and Computer Assisted Surgery 4(4), 321–330 (2008)