Noname manuscript No.
(will be inserted by the editor)

Formal Probabilistic Analysis of a Surgical Robot Control Algorithm with Different Virtual Fixtures

Muhammad Saad Ayub · Osman Hasan

Received: date / Accepted: date

Abstract Surgical robots are increasingly being used in operation theaters involving normal or laparoscopic surgeries. The working of these surgical robots is highly dependent on their control algorithms, which require very rigorous analysis to ensure their correct functionality due to the safety-crital nature of surgeries. Traditionally, safety of control algorithms is ensured by simulations but they provide incomplete 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, to verify the control algorithms of surgical robots in this paper. As an illustrative example, the paper provides a formal analysis of a virtual fixture control algorithm, implemented in a neuro-surgical robot, using the PRISM model checker. In particular, we provide a formal Discrete-Time Markov chain (DTMC) based model of the given control algorithm and its environment. This formal model is then analyzed for multiple virtual fixtures, like cubic, hexagonal and irregular shapes. This verification allowed us to discover new insights about the considered algorithm that allow us to design safer control algorithms.

Keywords Model Checking · Probabilistic Model Checking · PRISM · Surgical Robotics · Control Algorithm

1 Introduction

Robotic tools are being developed and used for surgical procedures as they offer ease for the surgeons and quick recovery for the patients. Surgical procedures, such as minimal-invasive surgery (MIS), where small incisions are done on the operating body for the penetration of surgical tool, are being increasingly done using robotic technology. Robotic arms, as opposed to human arms, have high degrees of dexterity and thus can be used to perform these surgeries with less margin of errors. These kinds of surgical procedures are generally used in neuro-endoscopy and treatment of brain tumor where precision is critical as even a slight error can damage brain tissues adjacent to the operating location.

Despite the extreme precision of surgical robots, these man-made machines bring in their own inaccuracies with them. 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* [22]. These issues may lead to life threatening situations, due to the safety-critical nature of human surgeries [20].

In order to cater for the out-of-boundary problems, the conventional approach is to test the robot by manually operating it. 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 complete verification. The other most commonly used analysis method for finding out-of-bound errors is computer simulation [36], 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, and thus most of the

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

times an incomplete analysis is done by leaving a significant number of test cases.

Formal verification methods [16], i.e., computer-based methods for mathematical analysis for systems, have been used to overcome the above-mentioned inaccuracy limitations for many hardware, software and physical systems. Due to their rigorous and mathematical verification approach, formal verification methods have become an essential part of the design phase of many industrial products [27]. The two mainstream formal verification methods are theorem proving [16] and model checking [15]. Deductive techniques are used in theorem proving to verify the relationship between the logical specification and implementation of the given system. The verification process may require explicit user guidance and thus can be quite tedious, especially when using the more expressive higher-order logic [16]. Model checking [4], on the other hand, is based on state-space exploration methods. The system under verification is mathematically modeled as an automata. This model is then used within a computer to automatically verify that it meets rigorous specifications of intended behavior [27].

In order to overcome the limitations of traditional analysis methods in ensuring safe control algorithms in surgical robots, we proposed a probabilistic model checking based methodology in [3]. For the verification of the considered algorithm, we used a cubic virtual fixture using the upper and lower boundaries for each axis of the cartesian plane. The control algorithm was implemented in the PRISM Model Checker and we verified the properties of Deadlock Freedom, Reachability, Out-of-Boundary and Collision Freeness. The results were obtained for different widths of the slow zone and the maximum amount of force applied. However, this work was limited to the cubic virtual fixture, which does not represent a real-world scenario as the areas of the humans bodies where the surgery needs to be performed, i.e., the virtual fixture, rarely has a perfect square shape. This paper is an extension of our previous attempt as we introduce real-world scenarios of Virtual fixtures using Markovian models. In addition to a cubic virtual fixture, we also use two other Virtual fixture models, i.e., a hexagonal Virtual Fixture and an irregular shaped Virtual Fixture. The probabilistic properties are then verified using the control algorithm for these Virtual Fixtures. The quantitative information provided by these probabilistic properties can play a vital role in designing safer and more performance efficient surgical robots. As an illustrative example, the paper also provides the formal probabilistic verification of a control algorithm for a neuro-mate robot [36] that is used to perform skull surgeries. In particular, deadlock freedom, reachability, out-of-boundary and collision freeness are verified for different configurations of Virtual Fixtures. Moreover, we have validated our results by conducting real experiments using the Al-Zahrawi surgical robot [17].

The rest of the paper is organized as follows: We present an overview of the related work in Section 2. Section 3 provides an overview of formal verification and a brief introduction to model checking and probabilistic model checking to allow the reader to understand these foundational concepts of the proposed approach. The virtual fixture control algorithm used in the surgical robots is also explained in this section. Section 4 describes the proposed verification methodology for the control algorithms of surgical robot. The necessary specifications used to verify the correctness of control algorithms and the critical properties and functions required to formalize the surgical robot control algorithms are also described. Section 5 explains our formalization of the control algorithm implemented in the neuro-mate robot that is used to perform skull-based surgeries along with the formalization of all the three Virtual Fixtures. In Section 6, the models from the previous section are verified using our proposed methodology. Section 7 describes the validation tests of our verification results using the Al-Zahrawi robot. Finally, Section 8 concludes the paper.

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 [26] used probabilistic model checking to verify the flexibility property of swarm robots in a collective foraging scenario. Kim et al. [21] 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. [33] verified the autonomous decision making system of a personal home robot using the SPIN model checker. Scherer et al. [32] 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 [24,34]. Lahijanian et al. [24] verified the probability of the robot reaching its destination via a safe path. Similarly, Fainekos et al. [7] addressed the problem of generating continuous trajectories for mobile robots while satisfying formulas in temporal logic using the NuSMV

model checker. Saberi at al [31] used the mCRL2 language [10] to create a formal model for a multi-robot system by creating different communicating processes and the Modal u-calculus [9] to formally specify the desired properties. The robot workspace is assumed to be a grid that is equally divided into two dimensional cells, whereas only one robot can occupy a single cell at a time. The authors then verified three important path planning properties, i.e., *deadlock freedom* to check that any robot will not get stuck in a particular cell, *collision-freeness* to check that the robots in the model will not collide with each other in any case and the *reachability* property to ensure that the robot will reach its desired destination within a finite number of steps.

Li [25] used the HOL4 theorem prover to verify the collision freeness property for collision-free motion planning algorithm (CFMC) of a dual-arm robot. Both the arms of the robot work simultaneously and the major risk in their operation is the collision of both arms with each other. The objective of the verification was to verify that the robotic arms follow an optimum path from the source to the destination without colliding. The original algorithm was found to have a semantic inconsistency and was modified to improve the search efficiency and planning. The revised algorithm was reverified and proved to be collision-free.

The formal verification of surgical robots has been barely touched as the complex dynamics and uncertainties involved in these physical systems, makes their formal verification task quite challenging. Bresolin at al [6] 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. The sequence task, required in the particular procedure, was broken down in subtasks and the workspace was partitioned into boundaries considering safe and dangerous regions. Based on this setup, the authors verified that the force of puncturing needle applied to the patient should always be less than a particular threshold and the position of puncturing needle should always remain within the target region. It was found that many simple assumptions taken during the procedure can result in large errors. The model was accordingly refined to cater for these assumptions using a new semantic of hybrid automata, i.e., Almost-As-Soon-As-Possible(AASAP) [35] and was verified to be better than the previous abstraction.

Recently, a formal modeling and verification approach for the virtual fixture control algorithm for a surgical robot has been reported in [22]. 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 [29] theorem prover. They showed that the algorithm is unsafe and modified it to satisfy safe operations. 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). Moreover, using the approach, given in [29], one can only verify the safety property and cannot find out the probability of safety, which is a quite useful parameter. 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.

This paper is an extended version of our earlier work on the probabilistic analysis of Virtual fixture based control algorithm [3]. In our earlier work [3], we formally modeled a control algorithm and a single Virtual Fixture based on the cartesian plane with upper and lower limits for each axis in the language of the PRISM model checker. Three modules, i.e., the Force module responsible for generating non deterministic components of force, the velocity module that detects the position of the tool and attenuates or stops the tool if it is near or at the boundary of the Virtual Fixture and the position module that updates the position of the tool based on the velocity, were created. Two tools were initialized and the modules were replicated for both of them. The properties of Deadlock Freedom, Reachability, Out-of-Boundary and Collision freeness were verified for the Model. The results showed that the probability of the tool crossing the boundary increases as the maximum force applied at the boundary edges of the virtual fixture increases. The algorithm was also verified on an actual surgical robot Al-Zahrawi. We are going to extend this analysis by verifying this control algorithm for hexagonal and irregular Virtual Fixtures in this paper.

3 Preliminaries

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

3.1 Probabilistic Model Checking

Probabilistic Model Checking [14] is used for the formal analysis of systems that exhibit random behavior and thus can be represented as Markov chains [23]. Model checking is an algorithmic technique in which the probabilistic state based model of the given system is developed and the quantitative properties involving probabilities are checked. The model checker rigorously ana-

lyzes the system, keeping in view the properties specified, and tries to capture the likelihood of every possible execution of the system [23,8].

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). DTMC is a state transition system in which the transitions between states are labeled with the probabilities whereas CTMC captures, not just the probability of making transitions between states, but also the delays incurred before making transitions. These random delays are represented using exponential probability distributions. MDPs are DTMCs with non-deterministic transitions and PTAs are CTMCs with non-deterministic transitions [27].

Once the Markovian model of the system under verification is finalized, 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 Markovian model and probabilistic 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 [28,23].

Many probabilistic model checking tools exist and each one excels in one or a set of application domains [28]. For example, INFAMY [11] is dedicated for model checking of infinite-state CTMCs and PARAM [12] for the parametric probabilistic model checking of DTMCs. PASS [13] and RAPTURE [18] model checkers are designed for analyzing the Markov decision processes only. The Fortuna [19] model checker computes maximum probabilistic reachability properties for PTAs and reward-bounds properties for (linearly) priced PTAs. PRISM [23], on the other hand, supports model checking for DTMCs, CTMCs, MDPs and PTAs. It is a generic tool and we found it quite suitable for our work

3.2 PRISM Model Checker

PRISM [23] 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.

PRISM also provides the facility of simulation. The state space can be explored automatically and manually. A PRISM user can define the number of steps and time for guided simulation and thus can simulate individual states manually. All these methods generate different simulation paths and these paths can be back tracked and stored in a text file.

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 [23].

3.3 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* [30]. With the aid of these virtual fixtures, the robot manipulator is guided to move in the specified region [36]. A surgeon describes the operating volume by a series of planes oriented and positioned in space. These planar boundaries are divided into the following three zones [22,36]: *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 freedom of robotic movement is somewhat restricted.

The surgical robot is usually attached to a manipulating tool with rigid mechanical linkage. As the surgeon exerts force on the manipulating tool, the robot senses these forces and performs the desired movement. This kind of interaction between the robot and the surgeon is called *cooperative control*. 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 [22]. To implement these movements, an admittance control circuit [20] is designed

that converts the sensed forces and torques to velocity through a multiplicative factor. The equation governing the control circuit is as follows

$$\dot{\overline{p}} = \frac{d\overline{p}}{dt} = K(\overline{p})G(\overline{f})\overline{f} \tag{1}$$

Where overbars indicate vectors and the dot ($\dot{}$) indicates a derivative with respect to time. p is the position and $\dot{p}$ 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 in cases where the surgeon desires to have flexibility to move rapidly and should be low when fine movements are required. K is the gain term used to impose motion constraints on the tool are as follows.

1. In the safe zone, K is taken as an identity matrix to allow free movements of the robot.
2. In the forbidden zone, K is taken as zero to stop further movements into the forbidden zone.
3. 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. Once the tool enters the slow region, 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 by using the following equation where $n1$ is unit normal to the boundary.

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

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

4 Proposed Methodology

It is desirable to have generic, scalable and fault tolerant virtual fixture control algorithms. The biggest safety concern in the context of virtual fixture control algorithms is the event associated with leaving the specified area of operation. The proposed formal verification methodology, caters for formally analyzing all of these requirements and is primarily composed of the following steps:

4.1 Modeling Control Algorithm in PRISM

A virtual fixture based control algorithm usually consists of a set of inputs, a translation block and a position update block. The translational block converts the input force from the surgeon to the robotic 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.

We propose to use the following generic steps to formally model any virtual fixture algorithm in the language of the PRISM model checker.

1. **Identify Modules:** The first modeling step is to identify the modules in the given algorithm. We usually associate a module with every block, i.e., surgeon force, velocity calculator and position update. These modules are implemented as Finite State Machines (FSM) with augmented probabilities and are executed in parallel by PRISM.
2. **Identify Input and Outputs:** Data sharing among various modules of the given control algorithm is done via global variables created in each module. We have to provide the data types and upper and lower limits for these variables.
3. **Initialization:** The variables are initialized in their respective modules.

4.2 Functional Verification using Simulation

Once the model is ready in PRISM, it is compiled to check for any implementation errors. The compilation also tells us if any state has unrealistic probabilities. Next, the model should be checked using the random and interactive simulator of PRISM model checker. The simulation, using random test vectors, often detects some critical errors in the implementation, which can be corrected in the model. The main aim of simulating the model is to trace any implementation flaws, before performing exhaustive and thus relatively time consuming formal verification.

4.3 Formal Verification

We propose to verify the following properties that every surgical robot should hold in addition to the out of boundary property for control algorithm by model checking.

1. **Deadlock Freedom:** Verifying that there is no deadlock in the model is one of the basic checks for any

robotic system. This check allows us to find any state or situation where a robotic system gets stuck and cannot come out of this cycle.
2. **Collision Freeness:** 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. Taking into account our model and implementation of the position module, this will compute the probability that the tools share a same grid point.
3. **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.
4. **Out-of-Boundary:** The most important aspect of any surgical robot is that is stays within the operable area all the 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.

It is a common occurrence to encounter state-space explosion problem during the verification phase. In this case, we propose to reduce the size of the model and thus the state-space by reducing the allowed values within the range of force applied by the surgeon and the size of the operable area.

5 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 plane, 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. Figure 1 shows an example of the virtual fixture configuration in the Cartesian plane with safe and forbidden zone boundaries.

The movement of the tool in the virtual fixture is modeled using a grid based approach. The block size of the grid is considered to be of 1 unit movement. The boundaries of the safe and the forbidden zones are then determined by the number of blocks on the grid. The distance of the tool from the boundary can also be determined by the number of blocks it is away from the boundary. The relationship of the grid blocks and the boundaries in our model is 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 module that is responsible for generating the force applied by the surgeon, the velocity translation module that converts the force applied by the surgeon into the tool velocity and the position module that updates the position of the tool according to the velocity applied. The velocity module 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.

5.1 Force Module

The force module captures the behavior of the interaction of the surgeon with the system, which includes the behavior details 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

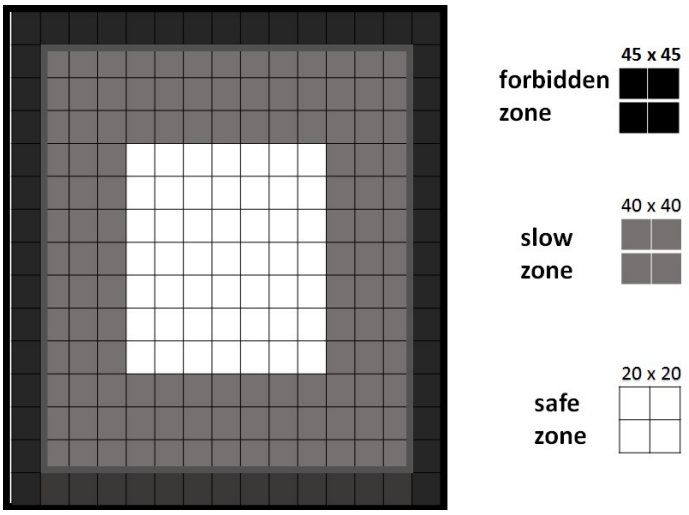


Fig. 1 Virtual Fixture Zones

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 [14], 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 and n represents the maximum grid size of the virtual fixture.

Pseudo code : Force

$1 : [](f = 0) \rightarrow 1/n : f = *;$
$2 : [](f > 0) \rightarrow 0.75 : f >= 0 + 0.25 : f < 0;$
$3 : [](f < 0) \rightarrow 0.25 : f > 0 + 0.75 : f <= 0;$

5.2 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, by any chance, crosses the boundary of the slow zone then the velocity is completely nullified and thus further movement towards the forbidden zone is prevented.

Pseudo code : Velocity

$1 : [](p > sl \quad \& \quad p <= sh) \rightarrow (v' = gain * f);$
$2 : [](p <= sl \quad \& \quad p > bl) \quad \& \quad f <= 0 \rightarrow (v' = (dl/Dl) * gain * f));$
$3 : [](p > sh \quad \& \quad p < bh) \quad \& \quad f >= 0 \rightarrow (v' = (dh/Dxh) * gain * f));$
$4 : [](p <= sl \quad \& \quad p > bl) \quad \& \quad f > 0 \rightarrow (v' = gain * f);$
$5 : [](p > sh \quad \& \quad p < bh) \quad \& \quad f < 0 \rightarrow (v' = gain * f);$
$6 : [](p <= bl \quad \& \quad f >= 0) \rightarrow (v' = gain * f);$
$7 : [](p <= bl \quad \& \quad f < 0) \rightarrow (v' = 0);$
$8 : [](p >= bh \quad \& \quad f > 0) \rightarrow (v' = 0);$
$9 : [](p >= bh \quad \& \quad f <= 0) \rightarrow (v' = gain * f);$

5.3 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.

Pseudo code : Position

$1 : [](p + v < n \quad \& \quad v > 0) \rightarrow (p' = P + v);$
$2 : [](px + v > -n \quad \& \quad v < 0) \rightarrow (p' = P + v);$
$3 : [](v = 0)|(p + v >= n)|(p + v <= (-n)) \rightarrow (p' = P);$

5.4 Model For 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 the tools in our model.

Pseudo code : Module Renaming

```
module forcex1=forcex [fx=fx1,a=b] endmodule
module forcey1=forcey [fy=fy1,a=b] endmodule
module velx1=velx [vx=vx1, a=b, ax=ax1, fx=fx1] endmodule
module vely1=vely [vy=vy1, a=b, ay=ay1, fy=fy1] endmodule
module positionx1=positionx [ax=ax1, vx=vx1, a=b] endmodule
module positiony1=positiony [ay=ay1, vy=vy1, a=b] endmodule
```

The modules are initialized such that both the 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. This is accomplished by creating a module `obstacle`, which is responsible for creating boundary points from the other tool's position.The variables ax1 and ay1 represent the new position of the tool computed from the velocity module.

Pseudo code : Obstacle Module

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

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 the 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}.

5.5 Model for Hexagonal Virtual Fixture

In real surgical procedures, involving surgical robots, the virtual fixture is usually not perfectly defined as a rectangular cube. It can be generalized to have any polygonal shape and the control algorithm must ensure that the tool remains within the given virtual fixture. This requires separate limits for each side of the virtual fixtures. In order to formally model these limits, we created a model with a hexagonal shaped virtual fixture shown in Figure 2. The virtual fixture is defined by the vertices of the hexagon both for the safe zone and the slow zone. The vertices are then used to find the widths of the safe zone and the slow zone by computing the distance from the origin to the midpoint of each side of the hexagon using the following equation:

$$distance = \frac{|\,(y2 - y1)x - (x2 - x1)y + x2y1 - y2x1\,|}{\sqrt{(y2 - y1)^2 + (x2 - x1)^2}} \tag{6}$$

Where $x1, y1$ are the coordinates of the start point of the boundary, $x2, y2$ are the coordinates of the end point of the boundary and x, y are the coordinates of the points from which to determine the perpendicular distance to the boundary. We implemented this equation in the PRISM model checker using the formula command. After determination of the safe and slow zones, the distance of the surgical tool from each boundary is computed using Equation 6. These distances are then compared to the widths of zones and force applied by the surgeon to determine if attenuation in the velocity is required. If the distance calculated is within the safe zone, the tool is allowed to move freely, on the other hand, if it is in the slow zone and the tool is moving towards the boundaries then the movement is attenuated. The velocity module was enhanced to increase the control algorithm for attenuating velocity as the tool approaches any boundary in the slow zone. The

module first checks the x and y components of force to determine in which direction the force is applied. The distance of tool from the boundaries in those directions are then checked. If the distance is greater than the width of the slow zone then the velocity is not attenuated, on the other hand, if it is less than the width of the slow zone then the velocity is attenuated. This ensures that the tool remains inside the virtual fixture at all times.

5.6 Model for Irregular Virtual Fixture

In the previous models, we defined the virtual fixture as a regular rectangle and a hexagon. Sometimes in real surgical procedures it is difficult to define a regular shape for operation area. An irregular polygon needs to be defined for the virtual fixture. This requires different boundary lengths to be defined for the virtual fixture. The virtual fixture is defined as shown in Figure 3. The distance of the tool in this scenario is also computed using Equation 6. The role of the sub-modules remains the same as described in the case of the previous model.

6 Formal Verification of the Virtual Fixture Control Algorithm using PRISM

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.

6.1 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.

```
E[F deadlock]
```

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 as this property was found to be failing, meaning that there is no deadlock in the model. The property was also checked for the control algorithm containing the hexagon and irregular virtual fixtures and no deadlock was found in either of the algorithms

6.2 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 specified as a probabilistic property as the result will not clearly depict if the tool reaches the boundary in minimal number of steps or not. On the other hand, by using the reward based approach we can ensure that the tool reaches 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]
```

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 4 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

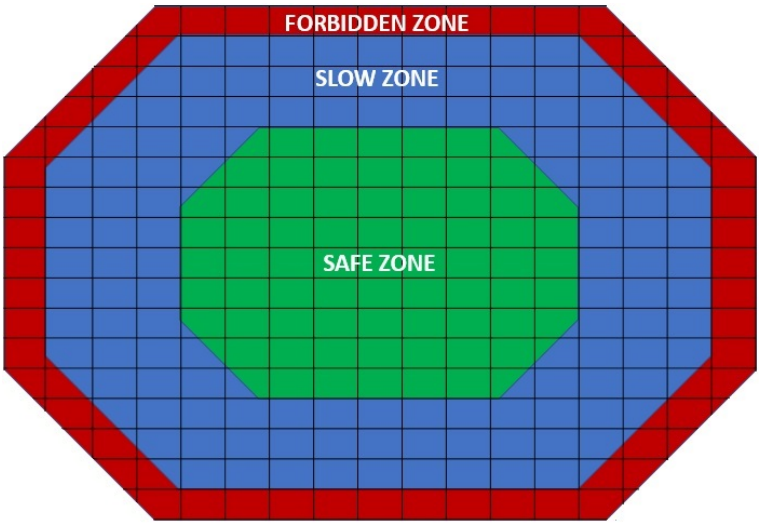


Fig. 2 Hexagonal Virtual Fixture Zones

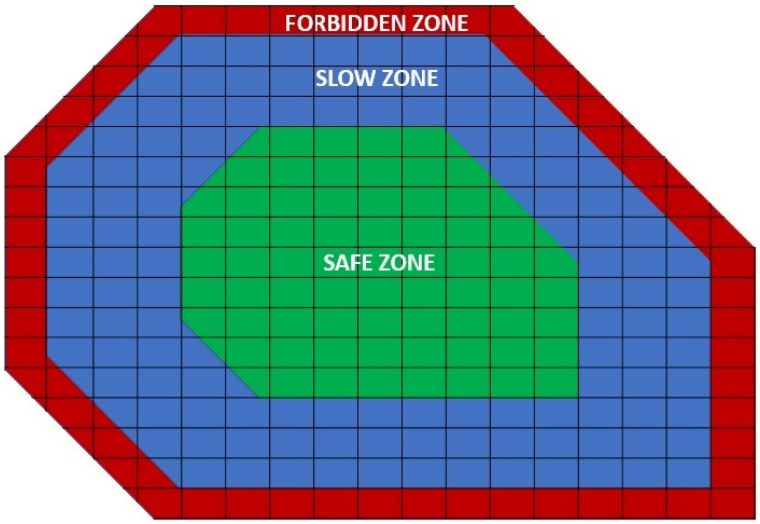


Fig. 3 Irregular Virtual Fixture Zones

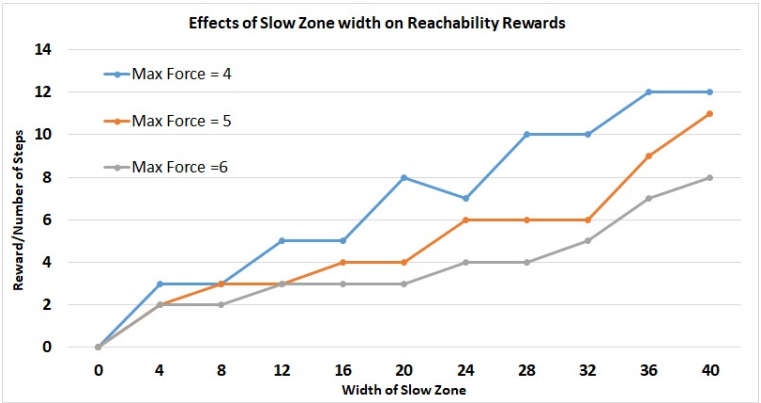


Fig. 4 Impact of Changing the Width of Slow Zone on the Reachability rewards vs width of slow zone property

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 4 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.

The reachability properties were also verified for the hexagon virtual fixture model for different widths of slow zones. As the position module is same, the properties defined previously can be used for the case of this model also. The results for the x-plane are shown in Figure 5. It can be seen that the rewards calculated are always finite as with the previous model. Also their value increases with the increase in the width of slow zone.

The result after testing the property for irregular virtual fixture model is shown in Figure 6. The rewards calculated are always finite.

6.3 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 that it stays within the operable area all the 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 7(a) shows the resultant probabilities computed after the verification of the above-mentioned properties at different slow zone widths and force limits. It is seen that when the width of slow zone is increased, the probability of the surgical tool crossing the boundary decreases (Figure 7(a)). The probabilities change by varying the maximum limit of force. 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 force 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 force 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 maximum force limits. This shows that the

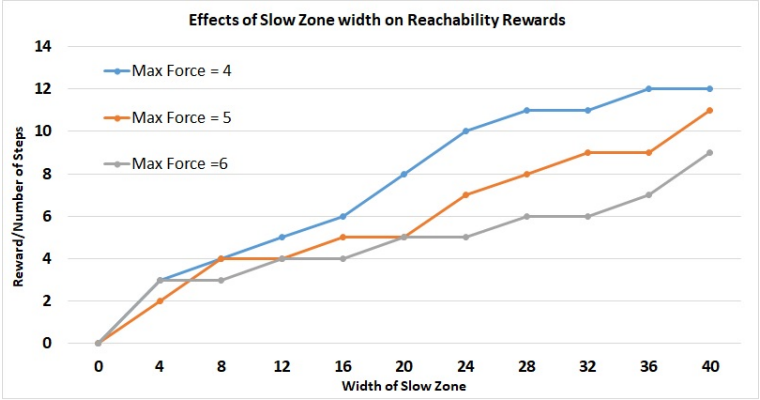


Fig. 5 Impact of Changing the Width of Slow Zone on the Reachability rewards vs width of slow zone property for Hexagon virtual fixture Model

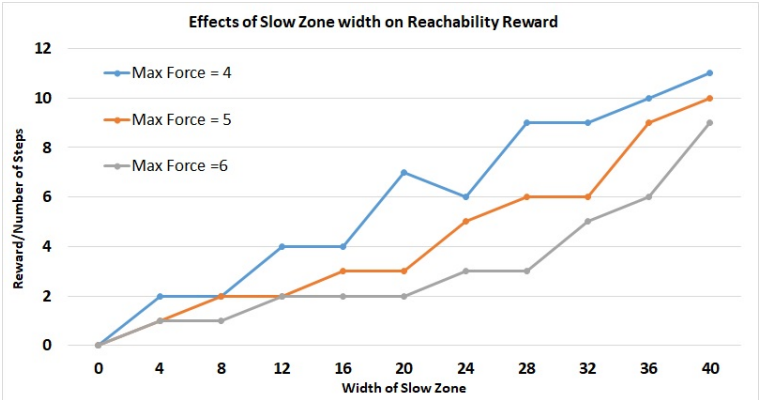


Fig. 6 Impact of Changing the Width of Slow Zone on the Reachability rewards vs width of slow zone property for Irregular virtual fixture Model

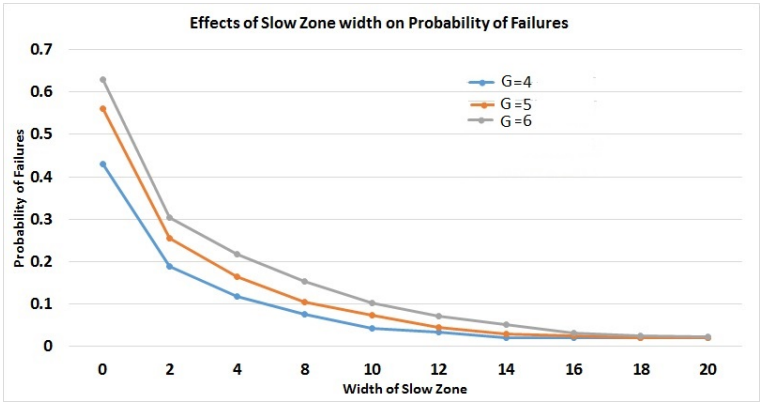
algorithm will not restrict the surgical tool from crossing the boundary if the surgeon exhibits sudden changes near the boundary of the forbidden region. The control algorithm needs to be updated to cater for these cases.

Figure 7(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 maximum and minimum force of 6 and -6 units, 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.

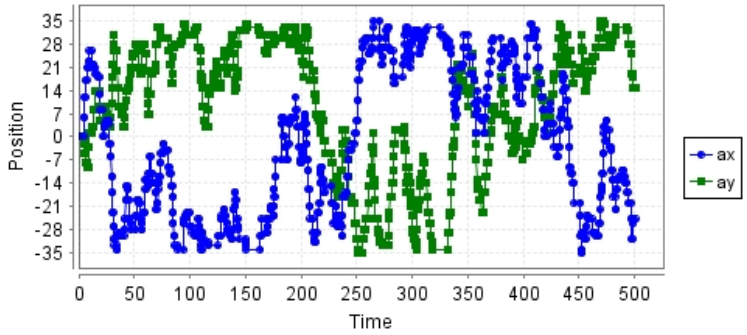
In order to verify that the tool will remain inside the hexagon virtual fixture, the property has to be updated to check the perpendicular distance of the tool from each of the boundary. The revised property is given as

```
P=? ((d1>0 & d1>dsafezone) | (d2>0 & d2>dsafezone)) &
   (fx>0) => F ((d1>dboundary1) | (d2>dboundary2))
```

In the above property, *d1* is the distance of tool from First boundary on the right side of virtual fixture, *dsafezone* is the width of safe zone, *d2* is the distance of second boundary on the right side of the virtual fixture, *dboundary1* is the distance of first boundary from the origin and *dboundary2* is the distance of second boundary from the origin. The above property computes the probability that the tool crosses the boundary of the virtual fixture when the distance of tool from the origin is greater than zero, the tool is not in the safe zone and x-axis force applied is towards the positive side. This property was checked for all the boundaries stated individually in both the directions. The results were computed at different widths of slow zones and at different values of the maximum force applied. The results



(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. 7 Results of Verification of Out-of-Boundary Properties

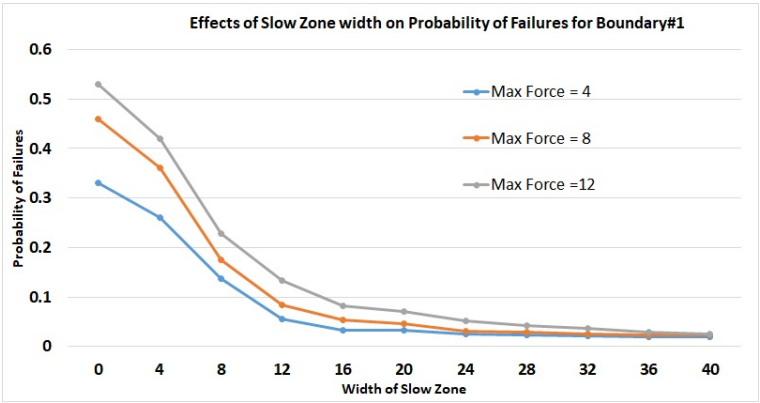


Fig. 8 Effects of the Slow Zone Width on Probability of Failures for Hexagon Virtual Fixture

were found to be quite similar to the ones obtained for the previous model, i.e., the tool crosses the boundaries when maximum force is applied at the boundary of the virtual fixture. Figure 8 shows the resultant probabilities for a single boundary. The out-of-boundary condition for the irregular virtual fixture model was verified using the properties defined in the previous section. The property specification was stated for all of the boundaries defined of the virtual fixture. The properties were verified at different widths of the slow zone and at different maximum forces applied by the surgeon. The verification results for one of the boundary is shown in Figure 9. The results show that as the width of slow zone increases, the probability of tool crossing the virtual fixture decreases. It can be concluded from the verification of the out-of-boundary properties that the behavior of the control algorithm is independent of the shape of the virtual fixture. Any irregular shape

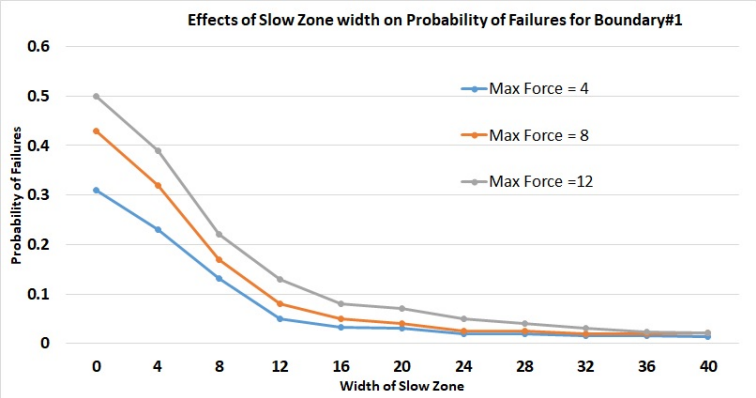


Fig. 9 Effects of the Slow Zone Width on Probability of Failures for Irregular virtual Fixture

with linear boundaries can be used as the virtual fixture.

6.4 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-plane, 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)
```

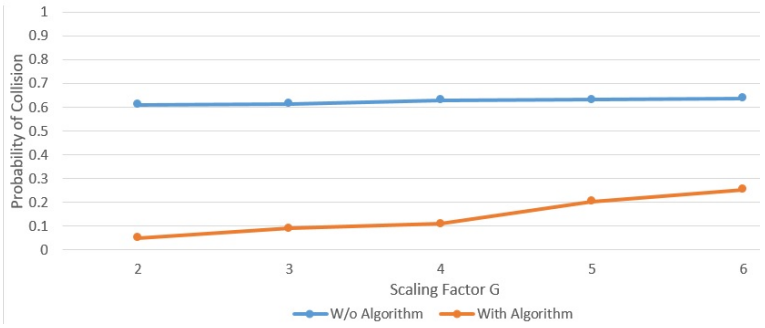
Where P is the probability of failure, $px1$ is the position of first tool, $px2$ is the position of second tool, sxh and sxl are the upper and lower limits of virtual fixture. This property is also checked for the y and z-plane.

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

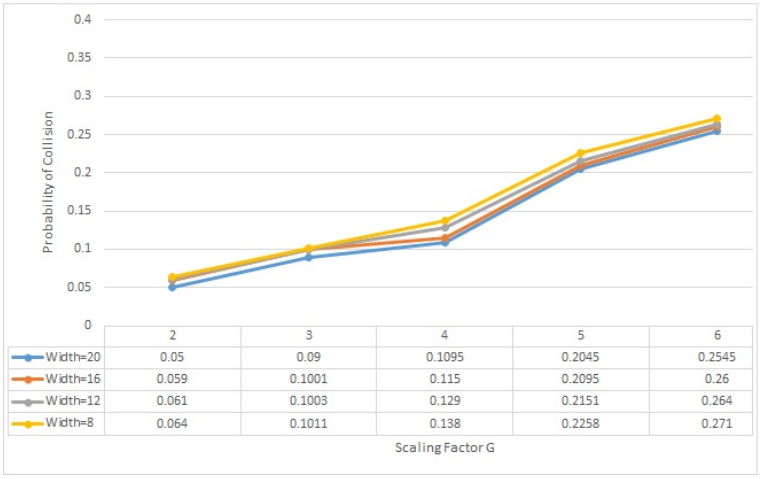
The size of the virtual fixture and the limits of force have a great impact on the verification of this property. In order to avoid the state-space explosion problem, the size of the virtual fixture is fixed and the maximum force applied is varied. These properties are verified for both the models, i.e.,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

Figure 10(a) shows the probabilities associated with the above-mentioned properties when verified for the model with and without obstacle avoidance algorithm at different force limits. The probability of collision for the model without algorithm remains almost constant by varying the maximum force limits as there is no restriction on collision and changing the force limits will not affect the collisions. On the other hand, for the case with the collision avoidance algorithm, it is observed that as the force limits increase the probability of collision increases (Figure 10(a)). This happens because, as the force limits increase, the tools become more likely to share the same grid point when the tools are near and a large amount of force is applied on one tool in the direction of the other.

Figure 10(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 10(b). It is observed that the width of slow zone does not affect



(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. 10 Results of Verification of Collision Freeness Properties

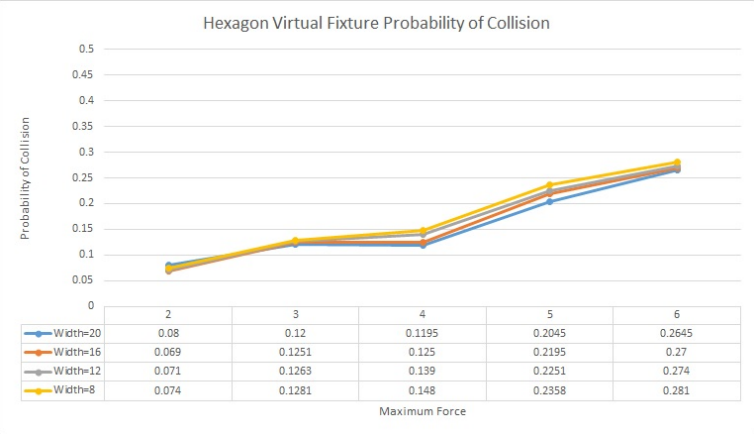
the collisions of tools and the probabilities of collision are almost the same.

Figure 11(a) shows the probabilities associated with the above-mentioned properties when verified for the hexagon virtual fixture model. It can be seen that the probabilities of collision for different widths of slow zone remain constant. Figure 11(b) shows the results after verification of the above-mentioned properties for irregular virtual fixture.

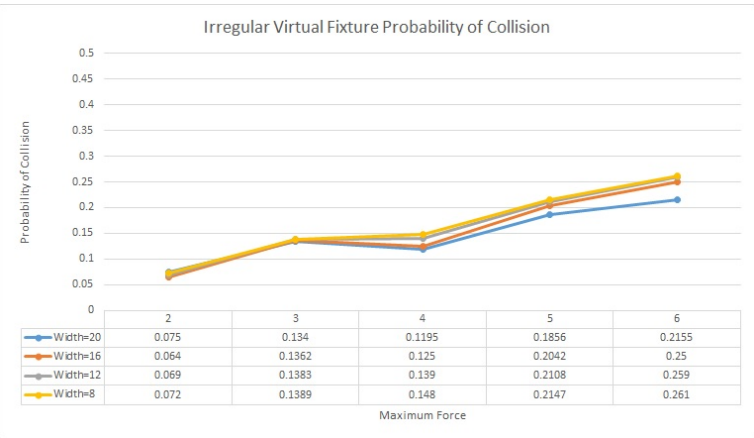
7 Testing on Al-Zahrawi

In order to validate our verification results, we tested the considered virtual fixture control algorithm on a minimal invasive surgery (MIS) robot Al-Zahrawi [17], 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 12. 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 12(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 12(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,

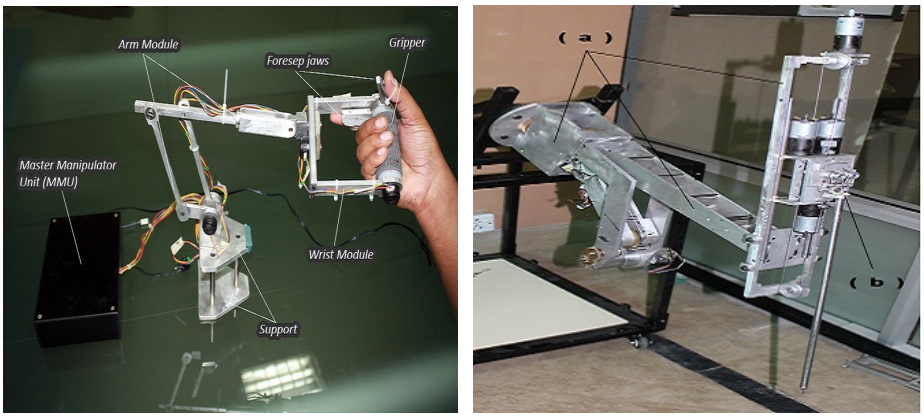


(a) **Effect of Force on Probability of Collision for Different Widths of the Slow Zone for Hexagon Virtual Fixture**



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

Fig. 11 Collision Freeness Properties



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

Fig. 12 Consoles of the Al-Zahrawi Surgical Robot [17]

one at the center and four at the boundaries of a rectangular region.

The slave console, shown in Figure 12(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.

7.1 Experimental Setup

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, shown in Figure 13, consists of five different positions, one at the center and four at the boundaries.

The experiment procedure can be partitioned into the following steps:

1. An object is placed at the center of the testbed.
2. The user is asked to pick that object using the telesurgical tool by operating the master manipulator.
3. The user is then asked to move the object to any edge box and try to place the object at the center of that box.
4. The boundary crossings in moving and placing the object are then recorded until the user places the object in the box.
5. The user then picks the object and places it in the box located at the opposite corner of the testbed.
6. The boundary crossings are again recorded until the user places the object in the box.
7. The above procedure is then repeated again by removing the virtual fixture algorithm and using a simple force to velocity translation.

40 different subjects, with various levels of expertise in robotic surgery, were asked to do the abovementioned procedure.

7.2 Results

The resultant boundary crossings of all the operators are logged and plotted in Figure 14 for both the cases, i.e., with the virtual fixture control algorithm and without the algorithm. The figure shows that the number of boundary crossings of the robotic tool without the algorithm are much greater than the ones of with the algorithm. It also shows that the tool crosses the boundary when it is operated 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.

8 Conclusions

This paper presents an extension to our previous work on the formal verification of a Virtual Fixture based control algorithm used in a surgical robot [3]. 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 three different Virtual fixtures, a control algorithm and also identifies the corresponding probabilistic properties. Our results confirm that the properties of out-of-boundary are failing but the probability of failure is very small, and thus it is quite safe to use the robot under these conditions for all the considered Virtual fixtures. Since traditional model checking cannot be used to verify probabilistic properties so these insights about the safe conditions cannot be obtained. This clearly indicates the usefulness of the proposed probabilistic model checking based approach.

We are considering to investigate enhancements in the virtual fixture based control algorithm to restrict the tool within the working zone while making the whole working zone reachable. Moreover, we also plan to do the formal probabilistic verification of other control algorithms, used in surgical robots, like active relative motion canceling (ARMC)[5], based on the proposed methodology.

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)

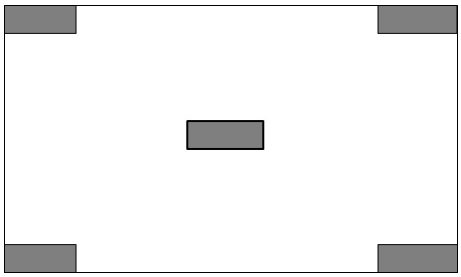


Fig. 13 Experiment Testbed for Al-Zahrawi surgical robot

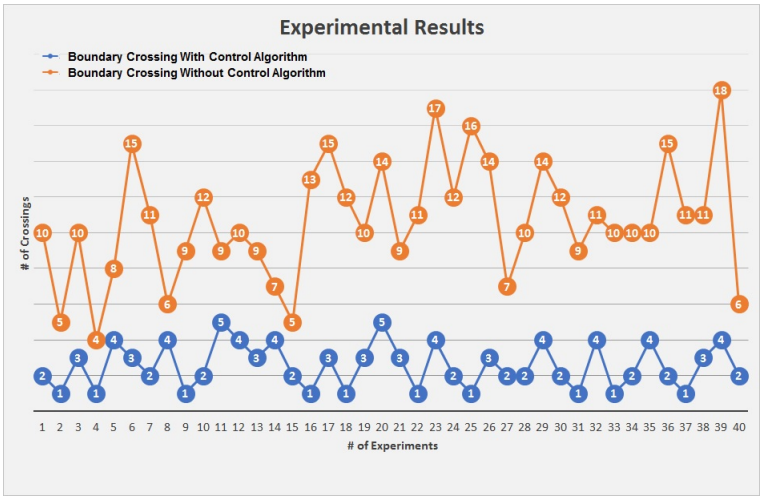


Fig. 14 Experimental Results

3. Ayub, M.S., Hasan, O.: Formal probabilistic analysis of a virtual fixture control algorithm for a surgical robot. In: Verification and Evaluation of Computer and Communication Systems(VECoS), pp. 1–16 (2017)
4. Baier, C., Katoen, J.P.: Principles of model checking. MIT press Cambridge (2008)
5. Bebek, O., Cavusoglu, M.C.: Intelligent control algorithms for robotic-assisted beating heart surgery. Robotics, IEEE Transactions on **23**(3), 468–480 (2007)
6. 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)
7. Fainekos, G.E., Gazit, H.K., Pappas, G.J.: Temporal logic motion planning for mobile robots. In: Robotics and Automation, pp. 2020–2025 (2005)
8. Grinstead, C.M., Snell, J.L.: Introduction to Probability. American Mathematical Soc. (1997)
9. 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)
10. Groote, J.F., Mathijssen, A., Reniers, M., Usenko, Y., Weerdenburg, M.V.: The formal specification language mCRL2. Citeseer (2007)
11. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Infamy: An infinite-state markov model checker. In: Computer Aided Verification, pp. 641–647 (2009)
12. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Param: A model checker for parametric markov models. In: Computer Aided Verification, pp. 660–664 (2010)
13. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Pass: Abstraction refinement for infinite probabilistic models. In: Tools and Algorithms for the Construction and Analysis of Systems, pp. 353–357 (2010)
14. 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)
15. Harrison, J.: Handbook of practical logic and automated reasoning. Cambridge University Press (2009)
16. Hasan, O., Tahar, S.: Formal Verification Methods. Encyclopedia of Information Science and Technology, IGI Global, pages 7162–7170 (2014)
17. 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)
18. Jeannet, B., Argenio, P.D., Larsen, K.: Rapture: A tool for verifying markov decision processes. In: Concurrency Theory (CONCUR), p. 149 (2002)
19. Jeannet, B., DArgenio, P., Larsen, K.: Fortuna: Model checking priced probabilistic timed automata. In: Quantitative Evaluation of Systems, pp. 273–281 (2010)
20. 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)
21. 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)

22. 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)
23. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification, pp. 585–591 (2011)
24. 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)
25. 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)
26. Mikaël, L.: Formal verification of flexibility in swarm robotics. Thesis, Department of Computer Science,Universit libre de Bruxelles (2012)
27. Norman, G., Parker, D.: Quantitative verification: Formal guarantees for timeliness, reliability and performance. Tech. rep., The London Mathematical Society and the Smith Institute (2014)
28. Oldenkamp, H.A.: Probabilistic model checking: A comparison of tools. Master's thesis, University of Twente, Enschede, Netherlands (2007)
29. Platzer, A., Quesel, J.D.: Keymaera: A hybrid theorem prover for hybrid systems (system description). In: Automated Reasoning, pp. 171–178. Springer (2008)
30. Rosenberg, L.B.: Virtual fixtures: Perceptual tools for telerobotic manipulation. In: Virtual Reality Annual International Symposium, pp. 76–82 (1993)
31. 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)
32. 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)
33. 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)
34. Whitcomb, L., Yoerger, D., Singh, H., Howland, J.: Advances in underwater robot vehicles for deep ocean exploration: Navigation, control, and survey operations. In: International Symposium on Robotics Research Navigation, Control and Survery Operations, pp. 346–353 (1999)
35. Wulf, D., M., Doyen, L., Raskin, J.F.: Almost asap semantics: From timed models to timed implementations. In: Hybrid Systems: Computation and Control, pp. 296–310. Springer (2004)
36. 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. Journal of Medical Robotics and Computer Assisted Surgery **4**(4), 321–330 (2008)