A Short Survey of Safe Reinforcement Learning

Nathaniel Hamilton
School of Electrical Engineering
and Computer Science
Vanderbilt University
Nashville, TN 37235, USA
nathaniel.p.hamilton@vanderbilt.edu

***Abstract*—Reinforcement learning is becoming extremely attractive for robotics, cyber-physical systems, and other safety-critical systems. However, the behavior of reinforcement learning systems is difficult to model. Thus, there are no guarantees that can be made about how the system will respond making it unsafe to implement in safety-critical systems. To combat this issue, Safe Reinforcement Learning methods and techniques are being researched and developed. This paper reviews and analyzes four papers with promising results from this field.**

I. Introduction

Reinforcement Learning (RL) is a branch of machine learning that focuses on software agents taking action in an environment to maximize rewards. The general idea is similar to training a dog to do tricks by giving it treats when it performs the desired task. Due to its generality and versatility, RL is also studied in disciplines outside of machine learning, such as game theory, control theory, simulation-based optimization, multi-agent systems, and swarm intelligence. RL is so versatile because it is a way of programming agents via reward and punishment without needing to specify how the task is to be achieved [1].

Reinforcement Learning and other learning-based methods have been around since the early days of computer science, but are rapidly gaining popularity in the control and artificial intelligence research communities [2]. The reason these methods are gaining so much popularity is because RL algorithms consistently show an ability to produce optimal results despite having poor models to work with. Because of this, RL techniques are extremely attractive for robotics applications, which have complex dynamics and environments that are difficult to model. However, the behavior of these RL systems are often difficult to interpret and predict. This is unacceptable for safety critical systems where unpredictable behavior could be the difference between life and death.

In order to avoid dangerous situations in safety critical systems, *Safe Reinforcement Learning* was developed. Safe Reinforcement Learning can be defined as the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes [3].

In this paper, we review four techniques developed in the last two years for safe reinforcement learning and how they relate to each other. Each technique focuses on safe exploration, but uses a different method of ensuring safety. The first two papers use runtime validation and temporal logic specification to detect unsafe actions and change them to safe actions before the action is processed by the environment. The third paper uses Lyapunov functions to determine a safe region that is updated and expanded as the learning agent explores the safe region learning more about its environment and the actual system dynamics. The fourth and final paper combines aspects from the previous three papers, developing a framework that prevents the learning agent from making unsafe actions while using the data it collects to update its environment and dynamics models.

II. Safety via a Monitor

Using a monitor to ensure safe and reliable behavior is a common technique used in real-time and control systems. A monitor is a third party agent that observes the behavior of the system and modifies actions when necessary. This is similar to a situation in driving where there is a second driver in the passenger seat who has their own set of brake pedals. This second driver observes the driving behavior and, when they feel unsafe, step on the brake to hopefully avoid the unsafe situation.

Monitors can perceive and detect unsafe situations in many ways, much like the second driver would depending on the person. The following two methods design their monitors to observe behaviors in different ways, but can be used with any type of learning algorithm.

A. Safe Reinforcement Learning via Formal Methods

The paper, [4], focuses primarily on using tools from the field of *formal verification*. Formal verification, the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property [5], provides a high degree of confidence that a system will operate safely. However, formal verification relies on accurate models of the system and even the best models are incomplete. For safety-critical systems, these small discrepancies could mean the difference between success and catastrophic failure. Conversely, RL-based controllers perform extremely well despite inaccurate models, but do not provide any safety guarantees. In [4] the authors aim to get the best of both worlds using a novel algorithm they call *Justified Speculative Control* (JSC). JSC combines off-line formal verification, runtime monitoring, and RL to transfer proofs of safety to learned policies.

The method described works by determining verifiably safe actions out of the current state and choosing one of the options from the list of safe actions. If no verifiably safe actions exist, the system is justified in taking a potentially unsafe, unverified action. Actions are determined safe using a verified runtime validation tool called *ModelPlex*. The algorithm for this method is shown in Algorithm 1.

Algorithm 1 Justified Speculative Learning

```
Require: $init, actions, environment, done,$
        $determine\_safe\_actions$
  $prev := curr := init$
  while $!done(curr)$ do
    $safe\_actions := determine\_safe\_actions(actions)$
    if $safe\_actions \neq \emptyset$ then
      $action := choose(safe\_actions)$
    else
      $action := choose(actions)$
    end if
    $prev := curr$
    $curr := environment(action, prev)$
  end while
```

Upon closer inspection of this method and algorithm, it can be seen that the monitor does not work the same way as described at the beginning of the section. It is similar, but fundamentally very different. Instead of the monitor only applying the breaks when necessary, this monitor design limits the possible actions the learner can take to only visit states it determines are safe. In the second driver analogy, this would mean that, instead of the second driver only having an extra set of brakes, this one has limiters that control how much the driver can turn, accelerate, or brake at any given moment. In the case where there are no possible safe actions, the monitor gives complete control to the learning algorithm to determine an action from the complete action space. When this happens, the learning algorithm is justified in making an unsafe action, hence the name *Justified* Speculative Control.

The authors validate JSC using three experiments involving simulated Adaptive Cruise Control (ACC) of one car following another. In the first experiment, the environment responds according to the model. In the second experiment, they introduce a slight perturbation to the relative position of the cars (2 units) with 5% probability. The third experiment demonstrates the effectiveness of a modified version of their algorithm that shows safer results when more error is introduced to the system model. The authors use Q-learning to determine their policies in the experiments, but note that JSC works for any generic RL algorithm.

In order to reduce the system state space, the authors use relative position instead of two distinct positions and limited the controller to three actions, brake with a constant force, accelerate with a constant force, or maintain current relative velocity. Additionally, the model is further simplified to only contain safe initial conditions. For example, the following car cannot start close to the lead car with a high enough relative velocity that even if the brake was constantly applied, the result would be a crash.

TABLE I
EXPERIMENT 1: JSC VS. CLASSICAL Q LEARNING IN A MODELED ENVIRONMENT

Training	JSC			Normal		
Steps	Crash	Fall Behind	Steady	Crash	Fall Behind	Steady
1,000	0	12559	289	10644	2162	42
10,000	0	12538	310	10462	2291	95
100,000	0	12375	473	10492	2284	72

TABLE II
EXPERIMENT 2: JSC VS. Q-LEARNING WITH ERROR INJECTION (.05 ERROR RATE)

Training	JSC			Normal		
Steps	Crash	Fall Behind	Steady	Crash	Fall Behind	Steady
1,000	3	12539	306	10950	1745	153
10,000	7	12502	339	10546	2215	87
100,000	5	12359	484	10561	2242	45

Experiment 1, JSC in an Accurately Modeled Environment: The results from this experiment, shown in Table 1, highlight JSC's ability to avoid unsafe states and improve the optimality of the controller. The non-JSC tests resulted in over $10,000$ crashes each, proving the method is unsafe. However, in all of the tests using JSC, no crashes occurred and the number of steady results were significantly higher than the non-JSC counterpart. A steady result occurs when the following car safely follows the preceding car without falling behind.

Experiment 2, JSC in an Environment that Admits Speculation: The results from this experiment, shown in Table 2, highlight JSC's ability to avoid unsafe states and improve the optimality of the controller even when the environment does not respond as modeled. While the JSC method does result in some crashes, the numbers are small; never reaching more than 10 even in the test with $100,000$ iterations.

B. Safe Reinforcement Learning via Shielding

The authors of [6] introduce a new type of monitor that enforces safety properties specified using temporal logic, a form of logic used for specifying and reasoning about propositions defined in terms of time. Their goal was to create a control system that combines the formal correctness guarantees provided by formal methods applied to temporal logic specification with the optimal performance, despite incomplete knowledge, provided by reinforcement learning. Using specifications defined with temporal logic, the authors synthesize a reactive system they call a *shield* to restrict the exploration space to only safe and recoverable states. They implement the shield two different ways, *Preemptive*

Shielding and *Post-Posed Shielding*[1], but restrict all of their testing to *Preemtive Shielding*. Therefore we will only focus on *Preemtive Shielding*.

The shield works the same way as the design from [4] except, with this design, there exist no cases where there are no safe actions. This is a direct result of how the shield is made.

In order to make the shield, the authors use game theory techniques to pit an abstract model of the environment against the safety conditions in a game that is run many times through simulation. In the game, the environment player chooses the next observations for the modeled state space, and the system chooses the next action. The environment player tries to force the system into unsafe states while the system player tries to prevent that. Through this game play a winning region, W, is generated. The winning region is a set of states and action pairs that will always lead to more states inside the winning region and never to unsafe states. The shield is then developed as a policy that allows all actions that are guaranteed to lead to a state in W, no matter what the next observation is. The shield essentially removes all traces that can lead to unsafe states. The result is, no trace allowed by the shield can lead to an unsafe state.

This paper validates their method through the use of four different experiments. Two involve a robot navigating a grid world and avoiding obstacles. One experiment uses the method to develop a control system for directing a car in an anti-clockwise square without crashing into the walls. Another experiment demonstrates the effect of using the method to play the Atari game *Seaquest^{TM}*. The last experiment simulates a heated water tank where the water level is controlled by turning on and off a valve.

In every experiment, except for the video game, the shielded system converges sooner and never reaches an unsafe state. In the video game experiment, the response was about the same with and without the shielding.

C. The Issue with these Monitors

While both methods have promising results and successfully prevent their experimental systems from reaching unsafe states, they have one major downside. In order for these methods to work effectively, they require accurate models of their environment for monitor design. In the method described in [4], crashes occurred when uncertainty was added to the environment and the method from [6] shows no results from added uncertainty. In order for these methods to be used in real systems, they must account for uncertainties and provide safety guarantees even when the environment does not respond as modeled. Therefore, while these methods work well in theory, they require further development before being tested in real safety critical systems.

[1]While *Post-Posed Shielding* is proven to be safe in the paper, the authors note it can train the learning agent to make unsafe decisions that rely on the shield to make the desired safe decision so the shield must remain after learning to ensure safety. However, *Post-Posed Shielding* can be used on learning algorithms that are already in the execution phase to ensure safety of unverified systems and the learning agent will likely not even realize the shield is there.

III. Maintaining Safety by Learning the Environment

Because RL algorithms provide optimal policies only in the long-term, intermediate policies can be unsafe, break the system, and/or harm the system environment. This can become even more of a problem for monitored algorithms because they are heavily dependent on accurate models of the dynamics. In this section, we look at a technique that works even when the dynamics are modeled poorly because the true dynamics are learned as the learning agent explores. Additionally, this method guarantees safety as it learns the true dynamics of the environment.

A. Safe Model-based Reinforcement Learning with Stability Guarantees

In order to combat unsafe intermediate policies for poorly modeled dynamics, the authors of [7] looked to *asymptotic stability* for a solution. More specifically, they focused on the *region of attraction* (ROA). The ROA is a subset of the state space that is forward invariant, meaning that any state trajectory that starts in the ROA, stays in the ROA and converges to a goal state eventually. Using this, the authors developed a novel algorithm that can optimize state action spaces while providing high-probability safety guarantees while demonstrating an ability to safely learn the dynamics of a system and increase the estimated safe region of attraction without ever leaving it.

Since the goal is to safely learn the dynamics from measurements and adapt the policy for optimal performance without encountering system failures, assumptions and restrictions are necessary. In order to generalize learned knowledge about the dynamics of the system at states that have not been visited, the authors restrict the system to being Lipschitz continuous.

For a function to be Lipchitz continuous, there must exist a real number such that, for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is not greater than this real number. In order to ensure the closed-loop system remains Lipschitz continuous, the authors also restrict the policies to the class of L_n-Lipschitz continuous functions, Π_L. Additionally, the system model must be statistically reliable in order to ensure the confidence intervals built on the dynamics cover the true function with high probability. When the system is adapting the policy, it is not allowed to decrease the ROA and exploratory actions are not allowed to drive the system outside of the ROA.

The general idea of how this method works is, the system starts with a small ROA based on the estimated dynamics of the system. The exploration of the learning algorithm is confined to that ROA, but free to explore anywhere inside of it. As the learning agent explores the ROA, the responses it receives are used to redefine the Lyapunov function that defines the ROA. As the function changes, the ROA expands

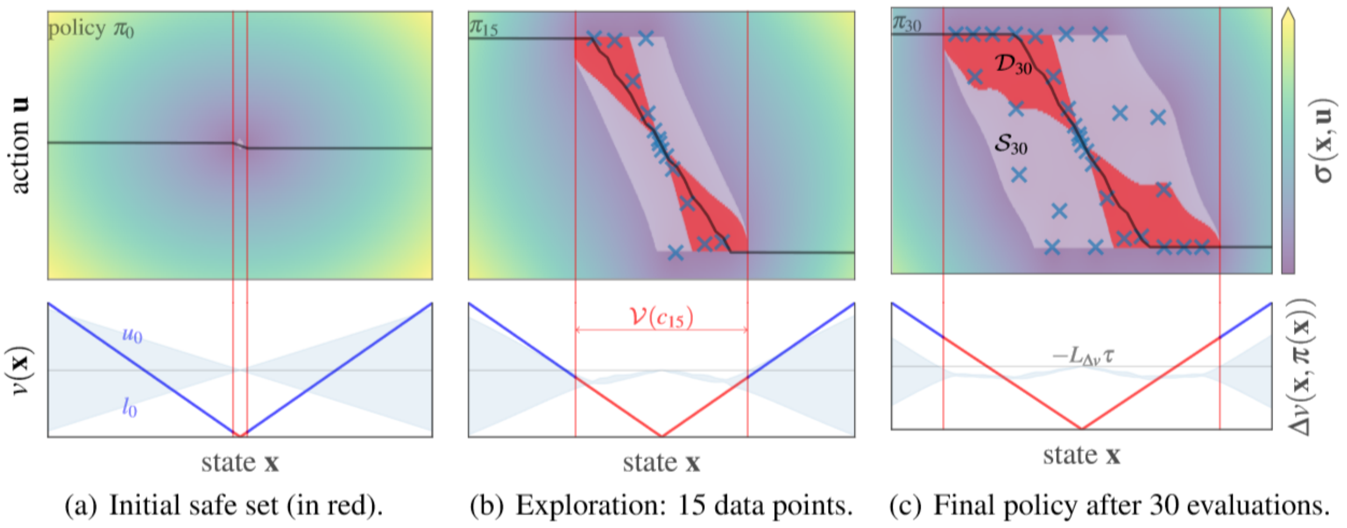

Fig. 1. Figure and following caption from [7]: We start from an initial, local policy π_0 that has a small, safe region of attraction (red lines) in Fig. 1(a). The algorithm selects safe, informative state-action pairs within S_n (top, white shaded), which can be evaluated without leaving the region of attraction $V(c_n)$ (red lines) of the current policy π_n. As we gather more data (blue crosses), the uncertainty in the model decreases (top, background) and we use a function to update the policy so that it lies within D_n (top, red shaded) and fullls the Lyapunov decrease condition. The algorithm converges to the largest safe set in Fig. 1(c). It improves the policy without evaluating unsafe state-action pairs and thereby without system failure.

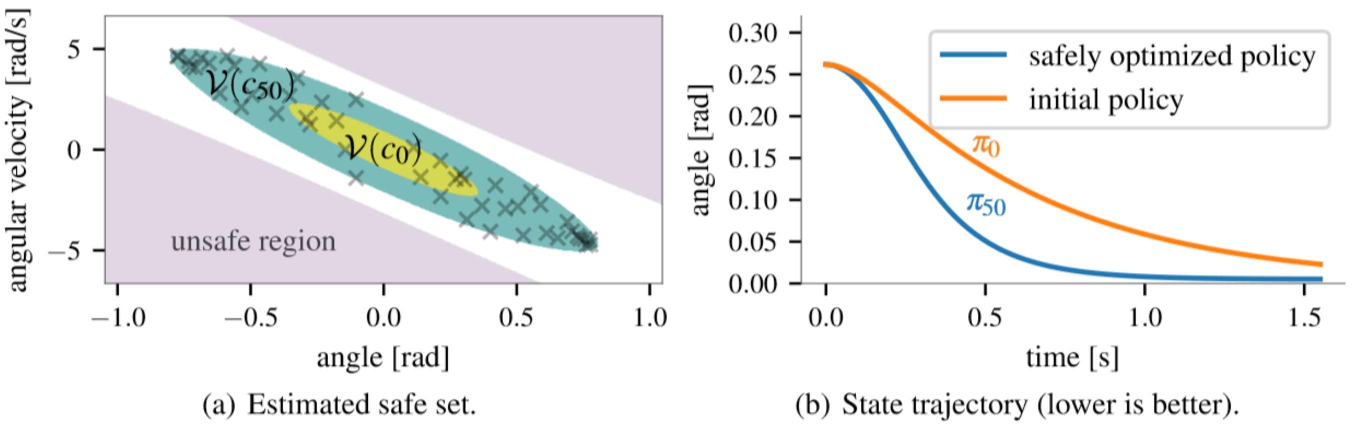

Fig. 2. Figure and following caption from [7]: Optimization results for an inverted pendulum. Fig. 2(a) shows the initial safe set (yellow) under the policy π_0, while the green region represents the estimated region of attraction under the optimized neural network policy. It is contained within the true region of attraction (white). Fig. 2(b) shows the improved performance of the safely learned policy over the policy for the prior model.

and provides more space for the learning agent to explore. This continues to happen until the ROA can no longer expand safely and the learning agent converges to an optimal policy. Thus the learning agent is able to learn an optimal policy without ever leaving the safe region. In other words, it maintains safety while learning the environment. This process is best illustrated in Figure 1, which shows how the ROA expands as the learning agent explores.

The authors demonstrate their method through an experiment with the classic inverted pendulum problem. For this they propose to use an approximate policy update that maximizes approximate performance while providing stability guarantees. It proceeds by optimizing the policy first and then computing the region of attraction for the new policy. They claim that this does not impact safety, since data is only collected inside the region of attraction and, should the optimization fail and the region of attraction decrease, they can always revert to the previous policy, which is guaranteed to be safe. They have to use an approximate policy update because part of their theory is intractable to solve. In doing so, they retain safety guarantees, but sacrifice exploration guarantees. However, this result is a more practical algorithm and the results are still promising. The results from their experiment are shown in Figure 2.

B. Issues with this Method

This method solves the issue of needing an accurate model of the environment in order to guarantee safety. The method is adaptive and flexible, learning the true dynamics of the environment. This is a marked improvement to the previous methods described in Section II. However, this method has some issues of its own that prevent it from being a solution for real life safety critical systems. When the system encounters large outside forces, there are no guarantees it can remain inside the safe region. As the agent is learning, there are no guarantees for situations when the agent is forced outside of the ROA by an outside force. Therefore, while this method works well in theory, it requires further development before being tested in real safety critical systems.

IV. Combining Both Techniques

Providing strong and practically useful safety guarantees for systems navigating unstructured environments requires more than model-based techniques, like the monitor, or data-driven techniques, like the learning strategy from Section III, can achieve on their own. Instead, both should be used in combination to achieve the best performance and safety guarantees. By incorporating learning techniques with a monitor, the system is able to ensure that it will never leave the safe region but also adapt and change the definition of that safe region if the environment dynamics are not accurate. In this section we look at a framework that combines both techniques to develop their own method that is tested in a real-world environment using drones.

A. A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems

The authors of [2][2] combine both techniques in their framework, which is based on Hamilton-Jacobi reachability methods and can be used with any arbitrary learning algorithm. As the system learns, they use a Bayesian mechanism to refine the safety analysis and determine new safety constraints. The result is a least-restrictive, safety preserving control law that only intervenes when necessary or there is low confidence in how the system will respond based on new observations.

The method works by implementing a monitor system based on the estimated dynamics of the system. This monitor invokes a safety controller whenever the system starts to leave the safe region, or safety envelope, or confidence in the estimated dynamics falls below certain threshold. In order to compute the safe region, the authors use Hamilton-Jacobi methods which calculate the worst-case scenario given the current estimated system dynamics. While the system is learning, they also implement an online Bayesian mechanism that refines the safety analysis as more information is learned about the environment. This reduces the conservativeness of the system while maintaining safe operation and strengthening the safety guarantees. Thus they ensure the system cannot leave the safety envelope while exploring and redefining the safety envelope.

Experiment 1, From Fall to Flight: The drone is initialized with a fully untrained policy, which will cause it to enter free fall as soon as it starts. This action will force the safety controller to take over and keep the drone within the safety envelope.

When the experiment starts, the drone plummets towards the ground because of the zeroed out feature weights for the controller. However, the safety controller starts before the drone is able to strike the ground. After that, the drone moves up and down very slightly as the control switches between the learning agent and the safety controller. As the switching occurs, the learning agent adapts its policy until sufficient feature weights are employed that allow the drone to follow the set trajectory. The results are shown in 3.

Experiment 2, When in Doubt: In this experiment the drone is initialized with ”hand-tuned” feature weights for the controller. The goal of this experiment is to generate new uncertainty bounds during the learning process. The results are shown in 4.

[2]Videos of the experiments and an additional explanation of the method can be found here: https://people.eecs.berkeley.edu/~jfisac/safelearning

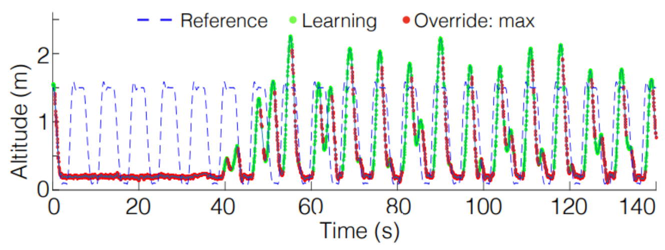

Fig. 3. Figure and following caption from [2]: Vehicle altitude and reference trajectory over time. Initial feedback gains are set to zero. When the learning controller (green) lets the vehicle drop, the safety control (red) takes over preventing a collision. Within a few seconds, the learned feedback gains allow rough trajectory tracking and are subsequently tuned as the vehicle attempts to minimize error.

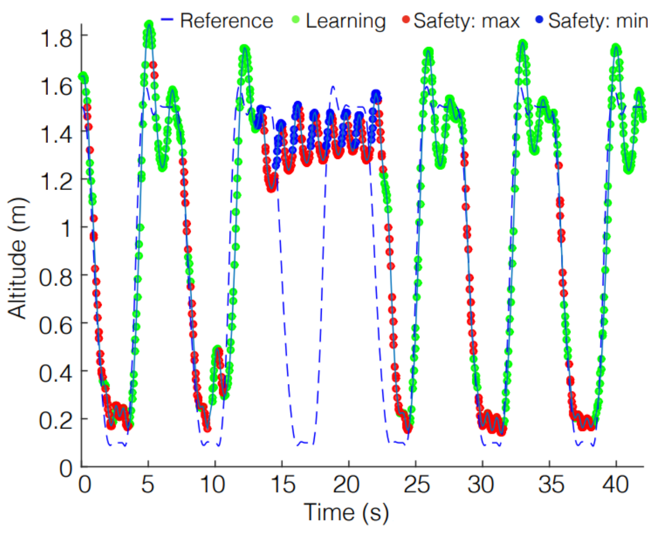

Fig. 4. Figure and following caption from [2]: Vehicle altitude and reference trajectory over time. After flying with an initial conservative model, the vehicle computes a first Gaussian process model of the disturbance with only a few data points, resulting in an insufficiently accurate bound. The safety policy detects the low confidence and refuses to follow the reference to low altitudes. Once a more accurate disturbance bound is computed, tracking is resumed, with a less restrictive safe set than the original one.

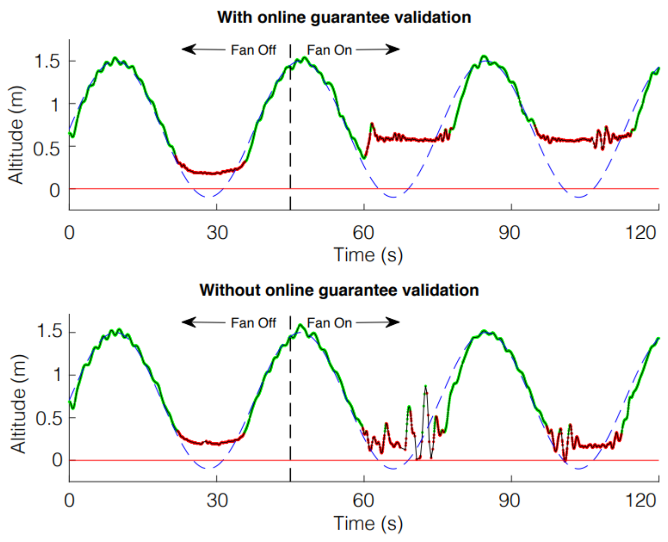

Fig. 5. Figure and following caption from [2]: Vehicle altitude and reference trajectory over time, shown with and without online model validation. After the fan is turned on, the vehicle checking local model reliability detects the inconsistency and overrides the learning controller, avoiding the region with unmodeled dynamics; the vehicle without model validation enters this region and collides with the ground multiple times. The behavior is repeated when the reference trajectory enters the perturbed region a second time.

When the experiment starts, the drone operates using the safety controller frequently. At $t = 10s$, the recorded disturbance values are used to compute a new disturbance bound, changing the safety guarantees and policy which are substituted into the drone's controls at approximately $t = 12s$. The new bounds are not received well as the drone repeatedly applies the safe control policy due to uncertainty. At $t = 20s$, the process is repeated, but with more data points. The new policy is less restrictive and the drone is able to continue following the prescribed flight path. According to the authors, after that point, the drone never switches to the safety controller due to uncertainty again.

Experiment 3, Gone with the Wind: In this experiment, the drone is initialized similarly to experiment 2. The flight path is the same, except this time the authors introduced a fan to create unmodeled disturbance. The purpose of this experiment is to test the effectiveness of their online validation and demonstrate the methods ability to respond to outside disturbances. As shown in 5, the drone with the online validation is able to continue flying despite the disturbance, while the drone without the online validation crashes to the floor twice. These results show that the system helps maintain safety conditions even when unmodeled interference is introduced.

B. Issues with this Method

This method successfully combines both techniques for great results. The method is very effective and proven to work on real systems, not just simulations. However, the method is computationally heavy, requiring an online component to handle a large part of the computations. The authors do not address what happens if the system loses connection to the online portion, meaning there could be a hole in their safety guarantee. They do mention work on reducing the computations and doing them on-board, but also note that it does not work as well. Therefore, while this is effective and very promising, it requires further work to improve its safety guarantees.

V. Future Work and Conclusions

While these methods and techniques worked well in simulation or a large empty room, the safety is only guaranteed for one agent. Additionally, the learning only applies to one agent. However, if these techniques and methods are to be implemented on autonomous cars or UAV's, the agents will have to interact with similar systems, learn in groups, and interact with humans. None of these methods account for that. Therefore, we propose that future work should try implementing these techniques and similar ones in a distributed setting[3], where agents are able to learn together and from each other. There are several open questions considered in future work including:

- Should there be one agent that handles the learning for all of the systems involved, or should they each learn on their own?
- If there is one learner, how does it generalize the policies to account for slight variations in the system dynamics of the involved systems?
- How can we transmit learned policies from one agent to another to share learning experiences, and is that a good idea?

Answering questions like these and more could lead to learning systems that interact with each other and inform others of how to respond in situations they have never encountered, thus reducing uncertainty and increasing safety.

References

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *CoRR*, vol. cs.AI/9605103, 1996. [Online]. Available: http://arxiv.org/abs/cs.AI/9605103

[2] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *CoRR*, vol. abs/1705.01292, 2017. [Online]. Available: http://arxiv.org/abs/1705.01292

[3] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, Jan. 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2789272.2886795

[4] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods," *AAAI*, 2018. [Online]. Available: http://nfulton.org/papers/aaai18.pdf

[5] A. Sanghavi, "What is formal verification?" *EE Times Asia*, May 2010. [Online]. Available: https://archive.eetasia.com/www.eetasia.com/STATIC/PDF/201005/EEOL_2010MAY21_EDA_TA_01.pdf?SOURCES=DOWNLOAD

[6] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," *CoRR*, vol. abs/1708.08611, 2017. [Online]. Available: http://arxiv.org/abs/1708.08611

[7] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe Model-based Reinforcement Learning with Stability Guarantees," *ArXiv e-prints*, May 2017.

[3]Some work has been completed in this area in [8], but more research is necessary.

[8] R. B. Larsen, A. Carron, and M. N. Zeilinger, “Safe learning for distributed systems with bounded uncertainties,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2536 – 2542, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317300873