GPS Spoofing Detection

Sakshi Namdeo* and Karthik Srivatsan†
Stanford University, Stanford, CA

Spoofing detection and mitigation is an ongoing research area in GPS integrity management. As Global Navigation Satellite Systems become more heavily used for autonomous navigation, many malicious actors find GPS signals a potential target for exploitation. Setting the problem up as a Markov Decision Process allows it to be tackled using a maximization-of-rewards procedure over all time steps. Our preliminary analysis shows results comparable to other machine learning methods.

I. Motivation

As autonomous robots become more and more commonplace, more stringent measures are needed to ensure that they can trust the information they receive. By sending fraudulent information to these robots, a malicious actor can deceive a receiver and take control. A common target of such "spoofing" attacks is GPS transmissions, due to their ubiquitous use in navigation.

Currently, some GPS receivers do have rudimentary defenses against spoofing: Receiver Autonomous Integrity Monitoring or RAIM can be used to detect unsophisticated attacks[1], and some modern GPS units are equipped with jamming-to-noise monitoring [2], which triggers an alert when the signal strength goes above a certain threshold. However, more sophisticated attacks can fool these rudimentary methods, leading any robot equipped with only a GPS sensor off course.

II. Literature Survey

Several researchers have tested GPS spoofing, sometimes with spectacular results. Psiaki [3] managed to spoof the GPS receiver on a yacht to make it believe it was traveling at supersonic speed several kilometers underground. Bhatti and Humphreys [4] tested a different approach, fooling a ship into believing it was on its correct, 200-m-wide path, while the ship was actually several hundred meters away. Their attack was designed to be as subtle as possible; when tested, they were successfully able to stay just under their own detection threshold. While this attack was tested in open water, it could damage a ship if attempted during a situation with tight maneuvering bounds.

Most GPS spoofers transmit signals that are stronger than authentic transmissions[5]. More sophisticated attacks try to lock on to the receiver correlation peak in order to fool it into synchronizing with the spoofer's clock. Choosing a valid signal based on Signal to Noise Ratio combined with a threshold level is used to counter these spoofing attacks. However, these decision rules are limited by attacks where the signal levels are slowly and adaptively increased. Many other attacks change the signal strength to keep it under the detection threshold of modern GPS units, or perform the spoofing over the course of minutes or hours to keep the predicted "integrity" high. A work-around for such a scenario is to employ supplementary sensors such as IMUs or use innovation sequences to keep track of positioning accuracy.

III. Methodology

GPS spoofing detection and mitigation can be represented as a POMDP, as defined in Kochenderfer [6], with components defined below. Parts of Atsushi Sakai's Python Robotics[7] toolkit was used, notably the EKF robot simulator. This file was heavily modified to fit our needs. Firstly, "spoofing" was added to the simulated GPS measurements. A "spoofing level"- the probability that a signal was spoofed- was chosen, and a random number generator was used to determine whether a particular GPS signal was spoofed. If it was, a position offset was added to the GPS signal. This offset grew as time increased, simulating a "drag-off" attack. Next, a simple proportional controller

*Electrical Engineering Department, Stanford University
†Aeronautics and Astronautics Department, Stanford University

was added to manipulate the robot's trajectory to maintain a desired path. The dead-reckoning functionality was also improved, by decreasing the time interval between successive evaluations of Euler's method.

To detect and mitigate spoofing, two key additions were made. The first was an observation probability function, computed using the intersection of two Gaussian distributions. The second was the reward function, which took in the robot's current state and the action it took, and computed a reward based on the position error each action would yield. Once the problem was defined in this manner, it was solved using the strategy described here.

A. Problem Definition

The problem is set up as an MDP by leveraging the position values received from GPS satellites and using an Extended Kalman Filter fitted within the robot. The EKF provides an estimate of the robot's trajectory given the GPS signals. An EKF is obtained by a linear approximation over non-linear systems and is widely used in tracking and localization applications.

1. States

There are only two states in this MDP: spoofed and non-spoofed. Spoofed indicates that the GPS signal is being altered by some malicious entity, and is not representative of the robot's true position. The amount of offset depends on the type of attack being simulated; most of the ones in this paper are slow drag-off type attacks. Non-spoofed indicates that the GPS signal is indicative of the true position of the robot, with some noise. Sakai's code uses Gaussian noise with a standard deviation of 0.5 meters. Our states are described as follows: $s \in \{0, 1\}$ where $s = 0$ means the signal is not spoofed and $s = 1$ means the signal is spoofed.

2. Actions

Given a potentially spoofed GPS signal, our robot can take one of two actions: trust or not trust. When the robot chooses to trust, it believes that the GPS data received is indicative of the true position. The observation then gets handed off to the EKF and is used for localization. When the robot chooses not to trust, it discards the GPS estimate, instead relying on dead-reckoning. Our actions are described as follows: $a \in \{0, 1\}$ where $a = 0$ means the robot decides not to trust and $a = 1$ means the robot decides to trust.

3. Transitions

Our transition function is encoded in the following table.

s'	s	a	$T(s'\|s, a)$
0	0	0	0.9
0	0	1	0.8
0	1	0	0.1
0	1	1	0.3
1	0	0	0.1
1	0	1	0.2
1	1	0	0.9
1	1	1	0.7

Table 1 Transition table

Here, s' is the next state, s is the current state and a is the action. In this table, we are encoding that transition to a different state other than our current state is not very likely. In an actual spoofing attack, the spoofer won't simply stop transmitting: therefore, regardless of our action, the transition from "spoofed" to "spoofed" is highly likely. It is also highly unlikely that we suddenly start getting spoofed; we have all used GPS for a long time, and have almost never

run into a spoofing attack at random. However, it is not completely impossible so our transition function has a slight probability associated with these outcomes.

4. Rewards

Cost is a function of state and action. Our costs are based on two factors, the current standard deviation of the GPS localization estimate and the current global truncation error (GTE).

s	a	$R(s,a)$
0	0	$-100GTE$
0	1	0
1	0	$-20GTE$
1	1	$-(\sigma_x + \sigma_y)$

Table 2 Reward table

These rewards were chosen due to the nature of the problem. Normally, GPS is used instead of dead reckoning to reduce navigational error; however, in a GPS-denied environment, both are needed to ensure good localization. The costs were chosen to reflect the impact each of these actions would have on the position estimate. In the case of "spoofed, don't trust", the cost is equal to 20 times the current GTE, since dead reckoning is used for estimation and it inherently drifts. In the case of "spoofed, trust", the cost is equal to the standard deviation of the estimate, since this action allows the spoofer to destabilize the robot. If the robot is not spoofed but it chooses not to trust the GPS signals, the cost is 100 times the GTE to penalize the robot for a wrong action. Finally, if the robot is not being spoofed, and it trusts the GPS signal, the cost is 0.

The weights in the "don't trust" cases were chosen to get the GTE on the same scale as the standard deviation, with the higher weight on the (not spoofed, don't trust) case reflecting that a sub-optimal action was chosen.

5. Observations

The robot observes two things: the current dead-reckoning position estimate and the incoming GPS signal. From this, it uses the probability function above to compute the likelihood of the GPS position estimate having the value it does given the dead reckoning estimate.

6. Belief

The probability of spoofing is computed by using an intersection of two Gaussian probability density functions; one that comes from the GPS signals received and the other from our EKF. The method then relies on dead-reckoning in times of spoofing to propagate the trajectory forward. The optimal policy is determined by passing the robot's belief state in to an estimated-reward computation, and finding the action that maximizes its utility given the belief state.

7. Policy

The policy adopted for this approach brings in the concepts of Value Iteration. The utility function associated with a particular action encodes information about the previous outcome, and is thus a one step greedy algorithm. The utility function is defined as:

$$U(a) = \sum_{s} p(s)R(s,a) + p(s)\sum_{s'}\sum_{s} T(s_{pred}|s,a)$$

where s_{pred} is our policy from the previous time step and p(s) is our belief of being in a particular state.

IV. Results and Discussions

To simulate a GPS-denied environment, we ran our Python code for varying levels of spoofing, drag-off attacks and simulation times. The results show that this algorithm is able to successfully detect our simulated spoofing with considerable accuracy.

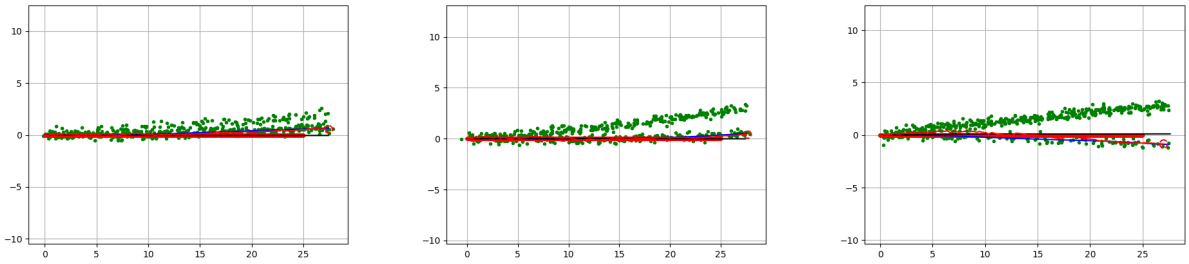

Fig. 1 Robot trajectory under different levels of spoofing $\in$ 0.25, 0.5, 0.75

The scattered green dots in Fig. 1 are the GPS Signals, the red line is our robot's estimated trajectory and the black line corresponds to the dead reckoning estimate. The ground truth is indicated by the blue line. The results come from a minimal drag off attack of 1 cm per spoofed signal. The simulation run time was 25 seconds with a GPS signal being observed every 50 milliseconds. The different levels of spoofing correspond to the percentage of received position updates that were altered. The robot's trajectory converges faster in the presence of a larger drag off attack, although the initial error in prediction is approximately 1 meter.

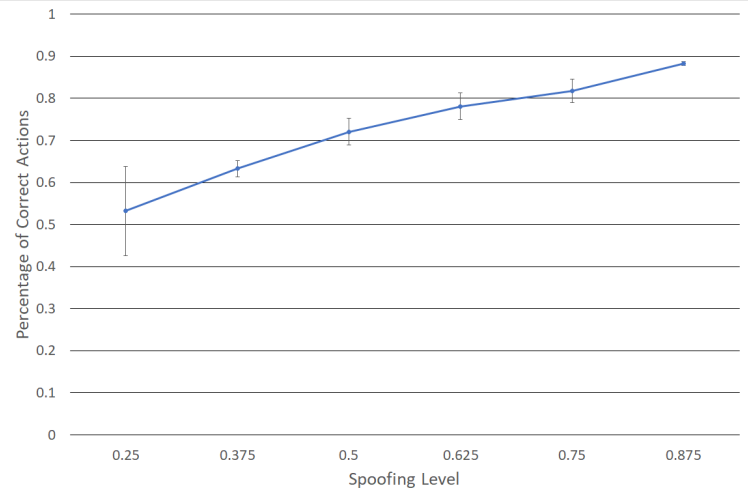

Fig. 2 Accuracy of Robot v/s Spoofing Levels

Fig. 2 reports the accuracy for the policy under different levels of spoofing. The accuracy shows an increasing trend as the amount of spoofing increases. This can be explained since larger deviations from our belief are easier to isolate.

Simulation results for our robot under invalid navigation signals showed adequate detection with our proposed policy. Certain interesting observations that are worth mentioning include:

- The convergence time for the robot is about 3 seconds for the attack shown in Figure 1. The time to converge decreases as the amount of displacement increases. This is likely because larger displacements are easier to classify as being malicious.
- As the level of GPS spoofing increases, the code generates better policies.

- For a drag-off attack, the accuracy of our policy grows as our simulation time increases. This directly ties into our assumption that a GPS signal is spoofed if the resulting variance in position estimates is large.
- Testing different trajectories showed that our results do not change when the desired trajectory is varied in a single dimension.

A. Localization

When anti-spoofing measures were not implemented, the EKF estimate of the position oscillated rapidly depending on the spoofing level. In several cases, the EKF estimate drifted several meters away from the true robot position. Due to this, the robot controller was fooled badly, often deviating 5 or more meters away from the desired trajectory. In the diagram below, a drag-off attack was performed on a robot with no anti-spoofing measures implemented. The spoofing level was 0.5, and it was maintained for 25 seconds. Over the course of this attack, the robot deviated by about one meter from the desired trajectory, with the EKF position estimate slowly being dragged away from the actual trajectory.

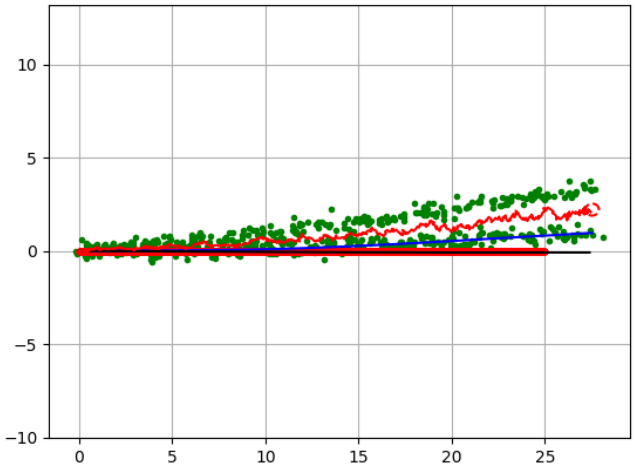

Fig. 3 Robot Trajectory with NO GPS Spoof Detector

The anti-spoofing measures described in this paper, however, proved quite successful. The drag-off attack described in Psiaki failed to fool our detection system, even with it altering nearly 90 percent of the incoming signals, as seen in Figure 2. The EKF estimate also remains much closer to the true position, despite some deviation in the beginning, as shown in Figure 1. The robot controller also kept the robot much closer to the desired path. There were several tests which ran the robot for 50 seconds: in these tests, the deviations from the robot not equipped with GPS spoofing detection only grew worse, while the ones with the spoofing detection were able to correct their deviations and get back onto the desired trajectory.

V. Conclusions & Future Work

A majority of the research in GPS Spoofing detection is to approach it using KNN classifiers or neural networks [8] which require a vast pool of data during the training phase. Collecting spoofed GPS signals in real-time is rather difficult and can be unlawful. Our results from this experiment solidify our belief that a POMDP-type approach can be formulated and solved for spoofing detection and mitigation. The results from our simulations corroborate the fact with our policy generating up to 90% accuracy for some attacks.

There are several potential extensions to this project. Here, we tested only a straight line and an arc: other trajectories could be tested, such as a circle or a zig-zag path with sharper turns. In this manner, we could determine if our methods actually are trajectory-invariant. The offset direction could also be changed, instead of being solely in the Y-axis, as it was in this report. The robot velocity could also be changed, instead of being kept around 1 m/s. By doing this, we

could find out whether that affects the accuracy at all. The problem could also be extended to three dimensions, to simulate aerial vehicles such as aircraft or drones that may come under attack from GPS spoofers.

Other spoofing attacks could also be deployed, such as those in Humphreys [9]. In this project, the attack used was a simple drag-off, which proved to be quite easy to detect. However, an attack that may prove much harder is the capture/drag-off attack, where the spoofer waits till it fools the robot's localization before it drags the robot off its planned trajectory. A combination of such attacks could also be simulated, to see whether it could throw off our detection algorithm.

Another avenue for exploration is generating actual GPS signals, using a GPS software-defined radio (SDR). In fact, we pursued this approach early in the project, believing it would provide the closest thing we could get legally to a real spoofing attack. However, the SDR packages proved to be very difficult to get working, and we moved on to our current approach. By using what we have developed in tandem with an SDR, we could determine whether our approach would work in a more realistic scenario, which takes in more features of a true GPS signal.

VI. Contributions

Sakshi Namdeo worked on writing the transition and observation functions, and suggested an approach using an EKF to find better estimates. Karthik Srivatsan worked on the robot controller and spoofing function, improved the dead reckoning, and generated the data visualizations. Both team members brainstormed the POMDP setup and solution, and wrote and edited all the sections of the final paper. All code can be found at: https://github.com/karthik-srivatsan/AA228-Final-Project

List of Tables

References

[1] Psiaki, M. L., and Humphreys, T. E., "GNSS spoofing and detection," *Proceedings of the IEEE*, Vol. 104, No. 6, 2016, pp. 1258–1270.

[2] Kerns, A. J., Shepard, D. P., Bhatti, J. A., and Humphreys, T. E., "Unmanned aircraft capture and control via GPS spoofing," *Journal of Field Robotics*, Vol. 31, No. 4, 2014, pp. 617–636.

[3] Psiaki, M. L., O'hanlon, B. W., Powell, S. P., Bhatti, J. A., Wesson, K. D., and Humphreys, T. E., "GNSS spoofing detection using two-antenna differential carrier phase," *Radionavigation Laboratory Conference Proceedings*, 2014.

[4] Bhatti, J., and Humphreys, T. E., "Hostile control of ships via false GPS signals: Demonstration and detection," *NAVIGATION: Journal of the Institute of Navigation*, Vol. 64, No. 1, 2017, pp. 51–66.

[5] Baziar, A., Moazedi, M., and Mosavi, M. R., "Analysis of single frequency GPS receiver under delay and combining spoofing algorithm," *Wireless personal communications*, Vol. 83, No. 3, 2015, pp. 1955–1970.

[6] Kochenderfer, M. J., *Decision making under uncertainty: theory and application*, MIT press, 2015.

[7] Sakai, A., Ingram, D., Dinius, J., Chawla, K., Raffin, A., and Paques, A., "PythonRobotics: a Python code collection of robotics algorithms," *arXiv preprint arXiv:1808.10703*, 2018.

[8] Shafiee, E., Mosavi, M., and Moazedi, M., "Detection of Spoofing Attack using Machine Learning based on Multi-Layer Neural Network in Single-Frequency GPS Receivers," *Journal of Navigation*, Vol. 71, 2017, pp. 1–20. https://doi.org/10.1017/S0373463317000558.

[9] Humphreys, T. E., Ledvina, B. M., Psiaki, M. L., O'Hanlon, B. W., and Kintner, P. M., "Assessing the spoofing threat: Development of a portable GPS civilian spoofer," *Radionavigation laboratory conference proceedings*, 2008.