Auto-encoding Robot State against Sensor Spoofing Attacks

Sean Rivera, Sofiane Lagraa, Antonio Ken Iannillo, Radu State
SnT, University of Luxembourg, Luxembourg
{sean.rivera, sofiane.lagraa, antonioken.iannillo, radu.state}@uni.lu

***Abstract*—In robotic systems, the physical world is highly coupled with cyberspace. New threats affect cyber-physical systems as they rely on several sensors to perform critical operations. The most sensitive targets are their location systems, where spoofing attacks can force robots to behave incorrectly. In this paper, we propose a novel anomaly detection approach for sensor spoofing attacks, based on an auto-encoder architecture. After initial training, the detection algorithm works directly on the compressed data by computing the reconstruction errors. We focus on spoofing attacks on Light Detection and Ranging (LiDAR) systems. We tested our anomaly detection approach against several types of spoofing attacks comparing four different compression rates for the auto-encoder. Our approach has a 99% True Positive rate and a 10% False Negative rate for the 83% compression rate. However, a compression rate of 41% could handle almost all of the same attacks while using half the data.**

***Index Terms*—anomaly detection, benchmark, robotic systems**

I. Introduction

Robots are cyber-physical systems, designed to perform specific tasks and ease human work. Current applications include, but are not limited to, military, industrial, agricultural, and domestic robots. With robots, the physical world is highly coupled with the cyberspace. Firstly, sensors perceive the physical environment; then, the control software chooses for actions, potentially in collaboration with other agents of the cyberspace (*e.g.,* other robots or the cloud); finally, actuators perform those actions on the physical environment. Regardless of their capabilities and size, **robots take their actions based on what they sense**.

Thus, robots are targeted by sensor spoofing attacks that can force an incorrect behavior in the robotic system and undermine the success and safety of critical operations [1]. Spoofing is the action of disguising a communication from an unknown source as being from a known one [2]. In this paper, we focus on Light Detection and Ranging (LiDAR) systems. These systems are largely employed to achieve high positioning resolution, in substitution or support of other approaches that locate the robot in indoor environments [3]. Any alteration of these sensor data can silently force the robot to initiate dangerous maneuvers for itself and the environment in which it operates in. Regardless of the purpose of the robot, sensor quality and robustness are highly requested to perform eventual mission- and safety-critical operations.

To detect anomalies in the LiDAR data, we propose a **novel anomaly detection approach for sensor spoofing attacks**. Our solution is based on an auto-encoder architecture, an artificial neural network which aims to learn efficient data representation in an unsupervised fashion [4]. Initially we train the model and we analyze the reconstruction error signals and create thresholds that will be used in the detection. In operational mode, the robot uses the encoder to compress the sensor input data. Then, the decoder is used to compute the reconstruction error and compare it with the thresholds for anomaly detection. This compressed data can also be efficiently sent to other robots or the fog/cloud for complex fingerprinting and overall monitoring. Unlike other auto-encoder solutions, we feed the neural network not only with the current data but also with past samples to detect changes more efficiently. Then, the auto-encoder architecture splits and processes different time windows to more efficiently detect changes. This approach adds a temporal correlation to the normal spatial correlation of our solution.

We implemented the anomaly detector and evaluated it against several types of spoofing attacks comparing four different compression rates for the auto-encoder. We first gathered the LiDAR sensor input data from two different simulations. Then, we duplicated them into several datasets and injected in each dataset a different spoofing attack at a random time. Our approach is effective with a 99% True Positive rate and a 10% False Negative rate for 83% space-saving, decreasing to a 50% False Negative rate while maintaining the 99% True Positive rate with the 11% space-saving. Detection is more effective with the 83% space-saving, however, the 41% space-saving could handle almost all of the same attacks while using half the data.

The contributions of this paper can be summarized as follows:

- a **novel anomaly detection approach for sensor spoofing attacks**, based on an auto-encoder solution that extract both temporal and spatial data correlations and enabler of efficient solutions with robot compressed state;
- a publicly available **collection of benchmark datasets** with the LiDAR sensor data injected with a state-of-the-art spoofing attack model.

Section II presents the related works; Section III formalizes the spoofing attack model used in this paper; the proposed approach is in Section IV; Section V presents the experimental results of our implementation; Section VI concludes the paper with limitation and future works.

II. Related Work

Robots with LiDAR systems have been proposed for remote surveillance of hazardous areas leveraging Simultaneous Localization and Mapping (SLAM) [5]. Robots with LiDAR systems are also used in smart environments as activity monitoring and health assessment [6].

However, robotic platforms suffer from security threats in the communication link and the applications, that are unique to them [7]. There are a number of relevant security flaws that can be used to take over, and command the robot. An attacker could stop the components that control the robot such as the camera, sensors, or legs to immobilize the robot [8]. Rivera *et al.* [9] proposed ROSploit, a modular offensive tool covering both reconnaissance and exploitation of ROS systems and sensors. Davidson *et al.* demonstrates sensor spoofing attacks [10] on robotic platforms. Petit *et al.* [11], presented remote attacks on camera and LiDAR of autonomous automated vehicles using commodity hardware, and another spoofing attack has been performed by Shin *et al.* [12], who proposed a spoofing attack causing the illusions to appear closer than the location of a spoofing device.

Software and hardware solutions have been proposed to countermeasure attacks on camera and LiDAR of automated vehicles [13]. Attack models consist of blinding, jamming, replay, relay, and spoofing attacks. Davidson *et al.* [10], proposed a spoofing attack against LiDAR and introduced a method for defending against such an attack on optical flow sensors, using the RANSAC [14] algorithm to synthesize sensor outputs. Choi *et al.* [15] proposed an attack detection framework that identifies external, physical attacks (including sensor attack, actuation signal attack, and parameter attack) against robotic vehicles on the fly by deriving and monitoring Control Invariants (CI). Kapoor *et al.* [16] detect spoofing attacks against an automotive radar system by effectively verifying physical signals in the analog domain. Rivera *et al.* [17] and Guo *et al.* [18] proposed robot intrusion detection systems that can partially detect sensor attacks.

In this paragraph, we present the research on anomaly detectors with autoencoders. Sakurada *et al.* [19] was the first that proposed the use of the reconstruction error of an autoencoder for anomaly. Similarly, Chong *et al.* [20] proposed using a deep learning approach to extract features from video frames. The autoencoders they presented are built with Convolutional Long Short-term Memory (ConvLSTM) model which is a variant of the LSTM [21] architecture. They experiment their approach on avenue, subway and pedestrians video datasets. Medel *et al.* [22] proposed a similar autoencoder solution by using ConvLSTM model on surveillance videos. The results were comparable to state of the art techniques. The authors evaluate their approach on the same dataset as [20].

Our work differs from the existing works mainly in how we process the input. We introduce both a temporal and spatial component for our correlation to detect a wide range of attacks in an efficient way. Furthermore, our anomaly detection approach can identify anomalies on sensors data incoming from robots such as LiDAR, that is different from image streams and other use cases in the literature. To the best of our knowledge, it is the first autoencoder solution specifically designed for analyzing the robot state. Furthermore, the proposed approach is also suitable for efficient communication of the robot state that can be used to create a global fingerprinting of the environment and enables more complex monitoring techniques with low latency.

III. Spoofing Attacks Model

In this section, we propose an attack model that targets a single sensor system of the robot. We chose to focus on the LiDAR sensor system as it is a common sensor type for robotics whose security we feel is underexplored. We extracted the model from the current state-of-the-art (section II) and the authors' experiences with robotic systems.

Independent Assumptions. We assume the robot will always start in an uncompromised state. The environment of the robot is not prone to sudden changes, such as objects appearing close to the robot without crossing the intermediate space.

We assume that the attacker can alter the sensor input state of the LiDAR system either through the associated software, directly interfering with the hardware, or broadcasting their LiDAR signals toward the receiver. We further assume that while the attacker can arbitrarily control the value of the LiDAR that the robot does not start under the attacker's control and that the robot has access to an unaltered LiDAR input.

Using these assumptions, we identified 10 potential spoofing attacks from both the state-of-the-art and the expertise of the authors.

Percentage Spoofing Attack. A percentage spoofing attack is one where all input data is shifted by a set percentage, multiplicatively changing all of the values with larger values more affected than smaller values.Building off of previous work [23], we choose to use a 10% change to all values, as it is the smallest percentage difference that can both ensure a robot collision and trap a robot with the default route planning software.

Value Spoofing Attack. Instead of spoofing by a percentage of the current returned value, the value spoofing attack shifted up or down the points by a set value of 5 centimeters (˜25% of the robots total size), introducing a predictable bias into the system that all objects are either nearer or further away.

Rotation Spoofing Attack. A rotation spoofing attack does not seek to change any of the values measured by the LiDAR sensor, instead of changing the indices where they are stored. Since these indices are the angles for the measurements, the effect is a rotation of the environment around the robot, such that the direction it thinks it is facing is no longer the direction it is truly facing.

Zero Replacement Spoofing Attack. A zero replacement spoofing attack simply replaces all inputs to a value of 0.

NaN Replacement Spoofing Attack. The NaN replacement spoofing attack replaces all values with a value of NaN (not a number). Unlike a Zero Replacement attack, this can occur

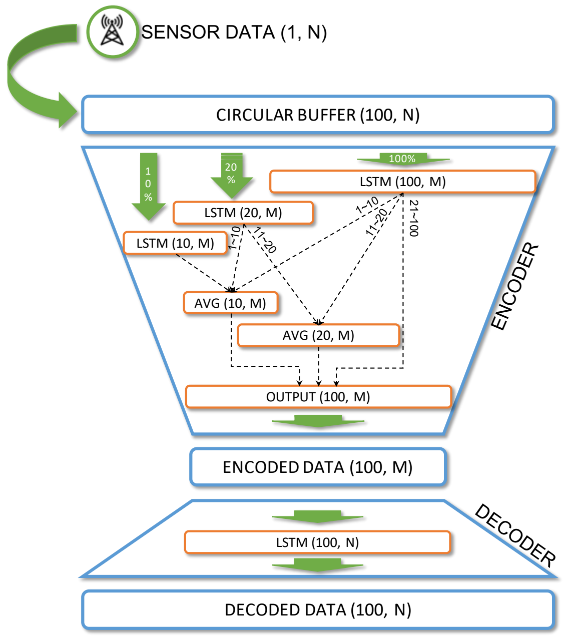

Fig. 1: Architecture of the encoder and decoder.

naturally if the robot is somewhere where the nearest obstacle is beyond the range of its sensor.

Repeated Data Spoofing Attack. The repeated data attack takes the last input to the robot and continuously replays it as the only new input with the addition of Gaussian noise similar to the generic sensor noise.

Window Repeated Data Spoofing Attack. In window repeated data spoofing attack, the data is repeated as a sliding window instead of having just 1 value repeated continuously.

Sector Value Spoofing Attack. Sector value spoofing attack aims to closely follow practical external attacks, where the attacker is not directly modifying the sensor data but instead is broadcasting their fake LiDAR signal from a direction.

Real World Spoofing Attack. The real world spoofing attack assumes that a LiDAR spoofer is a discrete object located near the robot that can only attack the parts of the LiDAR input that directly pass near it.

Frog Boiling Spoofing Attack. Named after the commonly known attack [24], the frog boiling spoofing attack attempts to avoid detection by only subtly changing the input at any point, below the detection filter. Over time, these small changes add up to a very large change in the input, but it is hard for the defender to see the change at any point.

IV. Proposed Approach

A. Architecture

The architecture of the detector is shown in Fig. 1. The detector is implemented as an LSTM autoencoder and it is split into two parts: an encoder and a decoder. The encoder is designed to create a compressed representation of the input data, while the decoder is designed to recreate the original input.

The input of the autoencoder is a circular buffer with shape 100xN floating-point values, where N is the dimension of the sensor input.This introduces a temporal correlation to the input allowing a memory of previous states At each timestep, the buffer is fed with another LiDAR measurement as a vector of N points, and the oldest value is removed from the vector. The input layer distributes the data over 3 LSTM cells to better detect spoofing attacks. The first cell works on only the past 10 samples and is the primary line of detection for the easiest attacks, which detection can be performed earlier. The second cell is trained on the past 20 samples and serves as the primary check for potential attacks. Since each attack effectively alters the input to the first layer twice as much as the second, this provides a clear measure of potential attacks. The last cell looks at all of the past 100 samples and is mostly used to detect slow or subtle attacks, such as Frog Boiling attacks.

The outputs of the LSTMs are merged in a stepwise manner, where the overlapping parts of each output are merged and the unique parts are appended. The output length of the LSTM cells can be tuned to alter the compression ratio and it is defined as M, where $M<N$.

The decoder component has its own LSTM cell, and it is trained to reverse the result from the encoder value and recreate the initial input. The output length of this last LSTM cell is N, the same dimensionality of the autoencoder input.

A necessary metric for our solution is the reconstruction error. The reconstruction error is a measure of how successful the autoencoder is at recreating the input once it has been encoded and decoded again. It is defined as:

$$reconstruction\_error = \Sigma(x-y)^2$$

where x is the autoencoder input and y is its output.

We measure the reconstruction error of the autoencoder to calculate the amount of noise in the system. Whenever the amount of noise vastly changes in the system it is a strong indicator that there is now spoofing in the system. We structure our system to strongly prefer false negatives to false positives. Unlike Sakurada *et al.* [19] we split the data into three separate LSTMs to provide a weighting of the input allowing for better pinpointing of the beginning of the attack.

B. Learning Phase

In the initial phase, we train the models over a attack-free dataset, to determine what is normal LiDAR data. We provide the model with examples of a wide variety of situations and allow it to learn the best way to represent the data in an unsupervised fashion. To train the neural network, the training data was read in and reshaped into a rolling set of 100 sample buffers offset by five samples each time, *i.e.,* the first buffer was sampled 1-100 while the second was sample 6-105. Once the model has been fully trained, the dataset is fed into it through the normal circular buffer one sample at a time. We record the decoded output and calculate the reconstruction error for each time step of the training data. Then, we determine the mean μ and standard deviation σ of the reconstruction error, which we store alongside the model for the compression phase.

C. Detection Phase

For the anomaly detection phase, the detector uses the encoded data and decodes them to reconstruct the original input. Then, it computes the reconstruction error and compares it to our previously computed mean μ. If the current reconstruction error differs from the mean μ more than $\pm 3\sigma$ [25], the alert of a potential attack is raised returning the sample wherein it is was detected. We believe our 3σ threshold is used to strongly discourage false positives in our calculations. False positives are a common source of issues within security systems and it is vital to limit this issue. [26]

V. Experimental Results

A. Setup

Our setup consists of a Linux Virtual Machine (Ubuntu 16.04) with 4 CPUs and 8 GiB of memory. The VM is emulated by qemu in a Linux Server equipped with Intel Xeon CPU E5-4650 v4 @2.20 GHz. We installed GAZEBO version 7.0.0 [27] for the simulation of robots and environments, ROS Kinetic Kame in the emulated robots, and our solutions have been implemented using the Python 3, Tensor Flow version 1.13.1 [28], and Keras version 2.2.4 [29] frameworks. The simulated robot is a Kobuki, equipped with an RPlidAR A2M8 as the LiDAR sensor. This sensor has a frequency of 6 Hz, *i.e.,* it reads 6 samples every second. A sample is a vector of 720 floating points that are the measurements from 720 directions around the robots, distributed in 8 sectors. Each point takes a value between 0 and 15 meters, that is the distance from the robot to the closest object in that direction. If the object is further than 15 meters, the value is set to NaN.The experiment is run using data collected from two different simulations built using the `rrt_exploration` packages for multi_robots [30]. In both simulations, five robots are tasked to explore the environment to build a shared common map without any truth reference. Thus, they rely on shared data to reconstruct the map. They differ in the simulated environments that the robots have to explore: the first is a one-story house, while the second is a larger maze.

B. Dataset Collection

The dataset is collected from the Gazebo simulations and stored in ROSbag files [31]. The data collection began when the robots started moving and concluded when all of the robots were confident that they had fully explored the area and correctly merged the map. The dataset contains many unique geometries for the LiDAR to record as well as multiple moving objects crossing in and out of vision.

Before we train the model, the dataset was normalized up to avoid any issue that could confuse or incorrectly train the model. *NaN* values, *i.e.,* nothing identified by the LiDAR, must be treated specially. All the points are normalized to a range between 0 and 1, but NaN values are substituted by the value 2 to indicate that the laser did not return and differentiate from actual points.

To inject the attacks in the datasets, we selected a random sample that was within 100 samples of its midpoint. The random value was chosen to ensure that there was no potential source of predictability for the testing. Once the starting sample was chosen, the attack was applied to the remainder of the data set. We created a total of 20 datasets for testing, two per attack type.

The datasets are publicly available at https://bit.ly/2Mndt4T. We strongly believe that sharing this benchmarks can help the research of robot anomaly detection by comparing different solutions on generic data.

C. Model Tuning

We instantiated four different detectors by tuning our architectural model. *N* and *M* are the two tunable values of the architecture, where *N* is the dimension of the inputs and *M* is the compressed data size (*cfr.* §IV-A). All four detectors work with data read from the same sensor, that provides a vector of points every timestep. Thus, we set *N* to 720. Every detector is characterized by a different value of *M*: 80, 150, 300, and 600. This provides different compression ratios that can be also computed as **state space saving**, *i.e.,* reduction in size relative to the uncompressed size. It is defined as

$$space\_saving = 1 - \frac{compressed\_data}{uncompressed\_data} = 1 - \frac{M}{N}$$

and it is expressed as a percentage. For example, the first detector is tuned with an output dimensionality of 80 for the encoder. It means that compressing 720 points to 80 saves the 89% of space for storing the robot state.

We trained the model on the dataset of the first simulation not compromised by any attack. We used the automatic split function in Keras to split the data 80% for training and 20% for testing, to train the autoencoder to pick up normal LiDAR behavior. Once the model has been trained fully, the data is run through a second time to calculate the mean and standard deviation of the reconstruction error for the model and the detection thresholds.

D. Experimental Results

We ran 80 experiments, one experiment for each dataset, simulation, and model. Results are shown in TABLE I. Latency is defined as the number of samples provided to the detector necessary for detection from the sample in which the attack started. Columns 2 to 6 report the latency for the experiments with the data from the first simulation, while columns 7 to 10 are for the second simulation. The second and third rows label the columns according to the compressed data size and state space-saving of the used detector.

Our solution works well in most of the situation (59 experiments out of 80), even with a high space-saving, detecting the attacks within an average of 20 samples of latency (3.33 seconds). In particular, our approach is effective with a 99% True Positive rate and a 10% False Negative rate for 17%

TABLE I: Detection Latency in Number of Samples (*ND*: not detected)

Dataset	First Simulation				Second Simulation			
Compressed Data Size (from 720)	80	150	300	600	80	150	300	600
State Space Saving	89%	79%	58%	17%	89%	79%	58%	17%
Percentage Spoofing Attack	*ND*	50	25	25	*ND*	75	25	25
Value Spoofing Attack	50	25	15	20	50	25	15	20
Rotation Attack	10	10	5	5	20	10	10	5
Zero Replacement Attack	10	10	10	10	10	10	10	10
NaN Replacement Attack	0	0	0	0	5	5	0	0
Repeated Data Attack	*ND*	*ND*	*ND*	*ND*	*ND*	*ND*	*ND*	*ND*
Repeated Data Window Attack	*ND*	*ND*	45	35	*ND*	45	35	25
Sector Spoofing Attack	10	10	5	5	50	20	20	10
Real World Spoofing Attack	*ND*	20	20	20	*ND*	50	20	20
Frog Boiling Attack	*ND*	*ND*	*ND*	85	*ND*	*ND*	*ND*	90

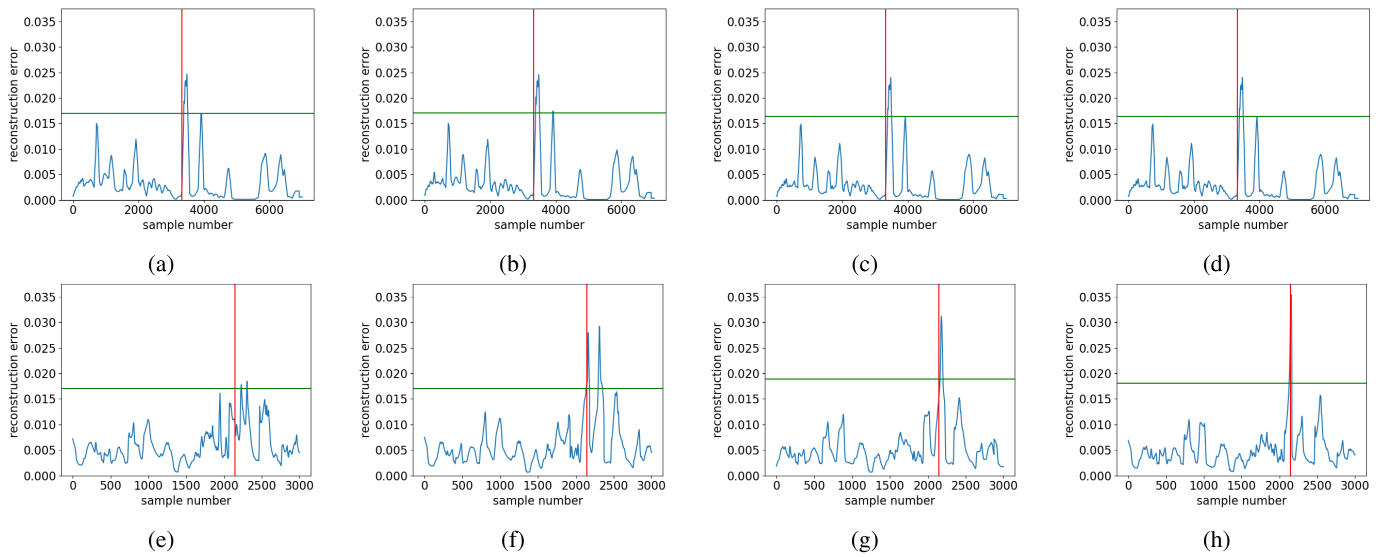

Fig. 2: Reconstruction error using the data from the first simulation (a,b,c,d) and second simulation (e,f,g,h), and with a space saving of 89% (a,e), 79% (b,f), 58% (c,g), and 17% (d,h). The horizontal axis is labelled with values that specify the sample number in the time series, while the vertical axis is labeled with construction error values. The red vertical line show the point where the attack starts, while the green horizontal line indicates the threshold for detection. Each pair of space saving has their own threshold for detection.

space-saving, decreasing to a 50% False Negative rate while maintaining the 99% True Positive rate for the 89% space-saving. Our True Positive rate is determined statistically, and is defined as a reconstruction error above our threshold when there is an attack. Our False Negative rate is defined as as a reconstruction error below our threshold when there is an attack. **Detection is more effective with 17% space-saving, however, 58% space-saving could handle almost all of the same attacks while using half the data.** Even if the detector was trained on the first simulation environment, it can detect the same attacks also on the new (and more complex) environment of the second simulation. The delay is a maximum of 40 samples (6.66 seconds) between the performances in the two environments at the highest compression rate and 10 samples (1.66 seconds) for the lowest. Figure 2 shows the computed reconstruction error over the course of the dataset for the value spoofing attack experiments. The figures clearly show the effect of the attack on the reconstruction error with the large spike of noise. The secondary spikes come from the three-layered approach as all parts of the detector react to the attack. Once the attack has filled the buffers, it is no longer 'novel' data and thus the error drops to normal. This means that once an alert has been raised the robot can no longer trust its LiDAR sensor until it has been returned to a safe state. The loss of information in the data compression prevents the detector to unveil a spoofing attack such as percentage, repeated data, real-world, and frog boiling spoofing attacks. In particular, the frog boiling spoofing attack is very challenging to detect, as the difference between any given two attacked samples is less than the difference from the noise between them. The proposed detection mechanism is not able to detect the repeated data attack because using only LiDAR data they are identical to the robot just stopping during normal operation. This scenario is often repeated within our data set, *e.g.,* the beginning of the operation, the end of the operation, and every time the map is being updated with new goals, and thus our detector assumes it is normal. This can be solved by adding meta-sensor information (*e.g.,* message time or last update) to the detector, training the detector to detect specific implementations of the attack, or by adding a new input from another source. If a different part of the robot thinks it is moving and the LiDAR is not changing, something has gone wrong. Overall, this system can detect the majority of attacks on the system, and allow for operator intervention.

VI. Conclusion and Future Work

In this paper, we proposed an anomaly detection method for detecting sensor spoofing attacks against robots. Our solution is based on an autoencoder neural network that leverages both the spatial and temporal features of the sensor data in order to reconstruct the robots sensor data and detect anomalies. Furthermore, we crafted datasets of compromised sensor data that emulate attacks on the LiDAR system and we made them publicly available as a benchmark. Our experimental results highlight the ability of our method to detect attacks. To the best of our knowledge, this is the first work who face the challenge of sensor data anomaly detection through autoencoders, by enabling also efficient robot state representation.

This work has also some limitations: usage of simulated data and limited experimentation. While we are pretty confident of the high quality of data provided by Gazebo, we understand that experimentation on-the-field will be more valuable. Furthermore, this work intended to be an exploratory research and we believe that the results are positive but we intend to perform a more complete design of experiments.

In the future, we plan to extend this work by using autoencoder as a hash function for sensor data. We want to build on this research by creating a fingering mechanism for robots such that distributed groups of robots can identify between them using their results, plus what the other robots have published. Ultimately we plan to build a framework for detecting security issues across heterogeneous groups of robots.

Acknowledgements

Research supported by EC H2020 Project CONCORDIA GA 830927.

References

[1] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling uavs with sensor input spoofing attacks," 2016.

[2] N. Hubballi and N. Tripathi, "An event based technique for detecting spoofed ip packets," *Journal of Information Security and Applications*, vol. 35, pp. 32 – 43, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212617301692

[3] "Amigo," https://robots.ros.org/amigo/, accessed: 2019-05-19.

[4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion."

[5] K. Y. V. Krishna, A. Wadnerkar, G. M. Patel, G. Baluni, A. K. Pandey, and R. M. Suresh Babu, "Indigenous Mobile Robot for Surveillance and Mapping," in *Machines, Mechanism and Robotics*, 8 2018, pp. 389–400.

[6] G. Wilson, C. Pereyda, N. Raghunath, G. de la Cruz, S. Goel, S. Nesaei, B. Minor, M. Schmitter-Edgecombe, M. E. Taylor, and D. J. Cook, "Robot-enabled support of daily activities in smart home environments," *Cognitive Systems Research*, vol. 54, pp. 258–272, 5 2019.

[7] K. M. Ahmad Yousef, A. AlMajali, S. A. Ghalyon, W. Dweik, and B. J. Mohd, "Analyzing cyber-physical threats on robotic platforms," *Sensors (Switzerland)*, vol. 18, no. 5, 5 2018.

[8] F. Martn, E. Soriano, and J. M. Caas, "Quantitative analysis of security in distributed robotic frameworks," *Robotics and Autonomous Systems*, vol. 100, pp. 95 – 107, 2018.

[9] S. Rivera, S. Lagraa, and R. State, "Rosploit: Cybersecurity tool for ROS," in *3rd IEEE International Conference on Robotic Computing, IRC 2019, Naples, Italy, February 25-27, 2019*, 2019, pp. 415–416.

[10] D. Davidson, H. Wu, R. Jellinek, T. Ristenpart, and V. Singh, "Controlling UAVs with Sensor Input Spoofing Attacks," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[11] J. Petit, B. Stottelaar, and M. Feiri, "Remote attacks on automated vehicles sensors : Experiments on camera and lidar," 2015.

[12] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds., 2017, pp. 445–467.

[13] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR," OnBoard Security Inc., Tech. Rep., 2015.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, pp. 381–395, 1981.

[15] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Xinyan, "Detecting Attacks Against Robotic Vehicles." New York, New York, USA: Association for Computing Machinery (ACM), 10 2018, pp. 801–816.

[16] P. Kapoor, A. Vora, and K. Kang, "Detecting and mitigating spoofing attack against an automotive radar," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–6.

[17] S. Rivera, S. Lagraa, C. Nita-Rotaru, S. Becker *et al.*, "Ros-defender: Sdn-based security policy enforcement for robotic applications," in *IEEE Workshop on the Internet of Safe Things, Co-located with IEEE Security and Privacy 2019*, 2019.

[18] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "Exploiting Physical Dynamics to Detect Actuator and Sensor Attacks in Mobile Robots," *arXiv*, 8 2017. [Online]. Available: http://arxiv.org/abs/1708.01834

[19] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction." [Online]. Available: http://doi.acm.org/10.1145/2689746.2689747

[20] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *Advances in Neural Networks - ISNN 2017 - 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21-26, 2017, Proceedings, Part II*, 2017, pp. 189–196.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, pp. 1735–1780.

[22] J. R. Medel and A. E. Savakis, "Anomaly detection in video using predictive convolutional long short-term memory networks," *CoRR*, vol. abs/1612.00390, 2016.

[23] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, 2016.

[24] E. Chan-Tin, D. Feldman, N. Hopper, and Y. Kim, "The frog-boiling attack: Limitations of anomaly detection for secure network coordinate systems." Springer.

[25] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*. Springer, 2017, pp. 189–196.

[26] A. T. Analytics, "New research from advanced threat analytics finds mssp incident responders overwhelmed by false-positive security alerts," Jun 2018. [Online]. Available: {https://prn.to/2uTiaK6}

[27] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator."

[28] TensorFlow, "Tensorflow," https://www.tensorflow.org/.

[29] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[30] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1396–1402.

[31] "Rosbag," http://wiki.ros.org/rosbag, accessed: 2018-03-27.