Learning Through Adverse Event for Collision Avoidance: A Self-Learning Approach

Hyunjun Han, Jusung Kang, Muhammad Asif Raza and Heung-No Lee*
Department of Electrical Engineering and Computer Science,
Gwangju Institute of Science and Technology (GIST)
Gwangju, Korea
e-mail: heugno@gist.ac.kr

Abstract—**We introduce a deep learning based collision avoidance based on learning events accompanied by an online, semi-supervised learning algorithm that allows the learning agent to gain experiences and learn by itself without any pre-acquired training dataset through online trial-and-error approach. Using distance sequences as inputs, two procedures are performed in the proposed algorithm; data gathering procedure and learning procedure. Simulation results show that our system can achieve minimum of 99.86% up to 99.99% accuracy in classifying collision event from all possible events, allowing autonomous agent to navigate within simulated environments without collision.**

Keywords-collision avoidance; deep learning; autonomous navigation; semi-supervised learning; autonomous agent; UAV

I. INTRODUCTION

While experiencing collision may be a negligible event to most beings, some objects are too fragile that even a slight collision could be catastrophic. One of such objects is a flying mobile robot with its rotating wings protruding its body, also known as Unmanned Aerial Vehicle (UAV). UAV must avoid from colliding into another object at all cost as the slight impact could severely damage its wings as well as the collided object.

Among recent researches regarding autonomous flight using deep learning techniques, one common approach is the use of image data acquired by cameras and processing them through Convolutional Neural Network [1]. This method is often accompanied by supervised learning to train their networks and produce outputs that specify actions agent needs to take when certain input image is given [2]–[5]. However, this procedure faces challenges in construction of image dataset itself due to large number of training sets required for properly training the networks, not to mention a time-consuming task that necessitates each training data be labelled beforehand. To solve such a data demanding issue, there have been researches regarding various methods of gathering or creating training data. Examples include self-supervised method in autonomously gathering collision image dataset [5] and use of synthetic 3D environment [6]. Still, large amount of dataset that must be pre-acquired and be labelled before training remains as a challenge yet to be solved in supervised learning.

Deep reinforcement learning is another method that is being researched in obstacle avoidance especially after recent progresses in reinforcement learning that have been proved to outperform human levels in certain fields [7]. Ranging from use of depth sensors [8], synthetic 3D images [6], to direct raw image inputs from cameras [9] as input data, DRL methods are based on trial-and-error procedures that optimize its performance in a given environment with proceeding iterations by taking actions that maximizes its future reward based on Q-value function. Thus, DRL is often data-intensive and time-consuming to train compared to supervised learning. However, the fact that no data labelling and pre-acquired training dataset are required gives DRL an advantage that supervised learning does not.

In this paper, we propose a new deep learning based collision avoidance system that allows agent to learn by itself and adapt to any environments using a single depth sensor in a simulated grid environment. We achieve this through a learning algorithm that combines advantages of both supervised learning and reinforcement learning; an online, semi-supervised learning that allows agent to by itself interact with its environment, gather data, and train throughout its learning procedure without requiring any pre-acquired training dataset. Through performance analysis under simulated environments, we show that the proposed learning method can effectively train the agent to avoid collision.

II. PROBLEM FORMULATION

The goal of this research is to make a learning agent using neural network that can navigate within a 2D simulated grid environment avoiding any collisions with no access to dataset required for training. No goals or destination points are considered in the simulation.

Under limited computation capacity, the agent is given only a single distance indicating sensor to interact with the environment, where the sensor is presumed to return precise distance measurement between itself and its facing obstacle without malfunction at any time input is required. Thus, we assume the agent has no knowledge on its surrounding environment. We assume that at every time-unit, the agent, or UAV, can only move forward, rotate left or right from its current position, and freely increase or decrease its travelling speed in a simulated grid environment.

III. Proposed Method

A. Model Selection

While collision is an event that is based on the distance between two or more objects, it also can be defined as an event that is dependent on time. That is, the same data containing distance information could be indicating an object is either moving closer to or moving away from its facing obstacle depending on its previous data. Hence, we modeled input to be a sequential data and adopted LSTM, or Long Short-Term Memory [10], which is a neural network architecture originated from Recurrent Neural Network (RNN) specialized in handling sequential data. Through LSTM network, we can effectively process sequential data, and omit the need to pre-process input data as to show changes in received data regarding time differences in a non-Markovian environment.

B. Input and Ouput Data Processing

The key factor determining the event collision is the distance between the moving object and the obstacle. As such, we define input from distance sensor as integer x_{ti} as a means to represent distance information, i.e., x_{ti} is a positive integer for $i=1, 2, \ldots, n$ where i represents distance measured at time t_i and n is the maximum sequence length. Hence, input data $\underline{x_{ti}}$ represents a distance between UAV and an obstacle along UAV's facing direction at time t_i, acquired by a single front facing depth sensor that returns distance value between itself and facing obstacle.

All experiences, or sequences of distance information acquired, will be stored into memory, or dynamic dataset, to be used for training as a supervised learning approach. However, when considering different travelling speeds of UAV, depending solely on the combinations of distance data to represent an event may produce multiple identical sequences of input with different outputs due to the simplicity of input data, i.e., a sequence of integers. Therefore, UAV's speed information x_{speed} has been additionally appended to input sequence as $(x_{t1}, x_{t2}, \ldots, x_{tn}, x_{speed})$ at the moment x_{tn} is acquired as to give uniqueness to each sequences regardless of different flight speeds. When n+1 numbers of input data from $(x_{t1}, x_{t2}, \ldots, x_{tn}, x_{speed})$ has been fed into an LSTM with maximum sequence length of n, a single event is formed.

An output is an indicator of either a collision or non-collision event depending on the state of agent which is defined by the validity of the current position. If current position of UAV after acquiring x_{tn} is out of bounds or has collided into an obstacle, state S_t is defined invalid and will produce output signaling collision. State is otherwise defined valid and will output non-collision for all the other happenings. Fig. 1 shows the overall LSTM model, input and output data processing when LSTM's sequence length is 4, i.e., n=4.

C. Learning Procedure

The goal of training is to optimize the given model to distinguish collision events from any other events, where each event is comprised of distance data acquired at different time along with additional UAV's speed information. By combining supervised learning and online learning approach, we have allowed the agent to dynamically gather data, store them, and learn through the accumulating data to train itself. The training procedure is composed of two phases, data gathering phase and training phase. Each phase switches back and forth between each other during the whole learning process.

1) Data Gathering Phase: In data gathering phase, the goal is to let agent experience, or collect, as many events as possible that represent either collision or non-collision through iterative trial-and-error procedure. Starting with an empty memory, the agent will record all experiences that it goes through into its memory along with definition on which experiences are good or bad depending on its current state at the moment of experiencing new event.

If the inputted sequence already exists in the dataset, it is directly processed through LSTM for an output and action will be taken accordingly as programmed that will prevent the agent from colliding into an obstacle. In this experiment, a random movement is selected when received event indicates non-collision. If, however, input indicates a collision, any other movement other than move forward is selected.

On the other hand, if input sequence is new, state of the agent is immediately checked. If current state is valid, the new input sequence will be assigned as a non-collision event. The input sequence and label will then be recorded into agent's memory, and the exploration within the map will be continued for further sequence acquisitions. If new input sequence results in an invalid state, this implies a discovery

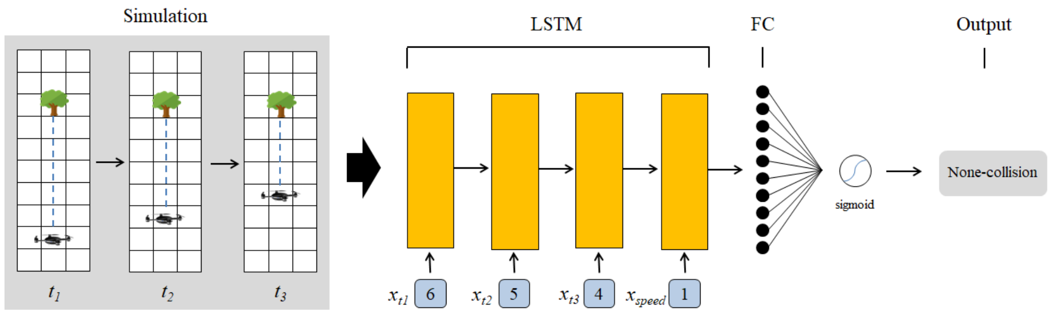

Figure 1. Agent at t_1, t_2, t_3, and its corresponding input to LSTM network, forming a input sequence of $\{x_{t1}, x_{t2}, x_{t3}, x_{speed}\}$

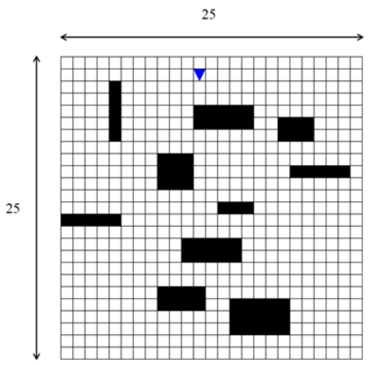

Figure 2. A flight environment with multiple obstacles (black squares) and agent (blue triangle) for simulation.

of a new collision sequence and will be labelled as a collision event. This new collision discovery shifts running process from data gathering phase into training phase.

Note that every time new input is acquired, the oldest input data, xt1, must be discarded while the new input takes the position of last time-step in the LSTM's input sequence for continuous experience acquisition.

2) Training Phase: In training phase, optimization of LSTM network is processed with the dataset that has been accumulated at its present memory, and happens every time a new collision sequence is discovered.

Once the training is complete with the provided dataset, phase shifts back from training phase to data gathering phase to further proceed in data gathering. The position of the UAV must also be reset to the initial position. As a result of the training phase, UAV in data gathering phase will be able to bypass learnt colliding sequences through LSTM processing, and continue its exploration into gathering new experiences.

IV. Simulation and Results

A. Flight Environmental Design

The flight environment for UAV has been designed as grid model with a total size of 25 x 25 grids. Multiple stationary obstacles of various sizes are randomly positioned within the grid map. Providing UAV a space to roam around and gather as many input sequences as possible is the primary purpose the flight environment must provide. Thus no complicated obstacle structures were additionally considered within the environmental settings. Fig. 2 illustrates a flight environment used for simulation.

B. Agent Settings

Agent, which we represent as a simulated UAV, can perform 3 actions while flying, i.e., rotate left, rotate right, and move forward. We have allowed diagonal flight within the environment by allowing each rotation to change the facing angle by 45 degrees, giving total of 8 different facing positions for UAV. A simulated depth sensor has been attached to the UAV's forward facing direction with the maximum distance measuring up to 20 grids. For every action UAV performs, distance measured from the distance indicator is read and used as an input to LSTM network. Lastly, the speed for the UAV is by default 1 grid per movement and may increase its speed up to 3 grids per movement, where each movement refers to a single time-step within the simulation.

C. Training

Training parameters were configured as follows; a single-layered many-to-one LSTM with 10 hidden nodes, Adam optimizer [11] with learning rate of 0.001 and mini-batch training with batch size of 128. With cross entropy as a loss function, sigmoid was used to output value between 0 and 1; 0 represents non-collision event and 1 represents collision. We used threshold value of 0.5 was used for classifying output value.

Before the UAV's first random movement, we have programmed it to rotate on its starting position one complete round as a means to acquire inputs to fully occupy LSTM with the sequence in order to prevent empty values from being processed through the network. Also, we have run the algorithm to increase its speed by 1 grid/movement after one complete run, allowing it to collect data at 3 different travelling speeds.

To show learning rate of our algorithm, we have first compared numbers of movements required before collision to number of iterations, or number of collisions learned. Fig. 3 shows total accumulated number of movements against the numbers of collision events learned. Despite randomness in learning agent's actions, the graph shows exponential increase in total movements required, and hence implies that more efforts to seek new experiences are required with respect to the amount of already experienced collisions.

D. Testing and Results

To test how well our trained model can classify collision and non-collision events, we constructed a testset by calculating all possible combinations of sequences that are available in the simulated environment, which numbered out to be 24,000, where 21,600 sequences are non-collision events and 2,400 are collision events. Note that the test set consisted of the trained sequences that had been gathered in

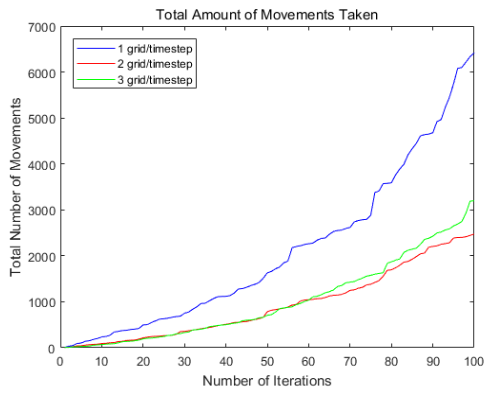

Figure. 3. Number of movements required before experiencing collision throughout training iterations.

the training process as well due to different number of dataset at each number of iterations. The average number of trained sequences after 100 iterations numbered 1770 for non-collision events and 300 for collision events, totaling 2070 learnt experiences, where each iteration is defined as learning a single collision sequence for specified travelling speed, i.e., 1 through 3 grid per time-step.

Using this test dataset, we first calculated precision (PPV), recall (TPR), and accuracy (ACC) for each iterations of learning processes. Precision, recall, and accuracy can be calculated as follows,

$$PPV = \frac{TP}{TP + FP}$$
$$TPR = \frac{TP}{TP + FN} \qquad (1)$$
$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP refers number of collisions correctly classified as collision, FN is collision incorrectly classified as non-collision, FP is number of non-collisions incorrectly classified as collision, and TN is number of non-collision events correctly classified as non-collisions. We have run the same experiments, or trials, under the exact same learning parameters 3 times to acquire each metric. Table 1 shows the result of each graph after reaching 50 iterations for trial 1, 2, and 3.

The highest recall and accuracy rate was achieved with trial 2, achieving recall rate of 99.96% at 43rd iteration and accuracy of 99.99% at 44th iteration. In case of precision, each trial could converge to 100% at least by 46th iteration. This means that our system misclassified only 2 events out of 24000 total events at the best within the three trials. The least number of misclassification was numbered 37, 2, and 21 accordingly for trial 1, 2 and 3 within the first 50 iterations.

V. Conclusion

In this paper, we proposed a collision avoidance system that is based on learning events where each event is composed of sequences of distance data acquired from a single simulated distance indicator. Using a semi-supervised training process that involves dynamic dataset by combining online supervised learning, we allowed the agent to gather data and learn by itself throughout proceeding training iterations without pre-acquired training dataset.

The experiment result showed that the proposed learning method allows agent to perform as good as full supervised learning method with full dataset given, achieving up to 99.99% accuracy in identifying collision among all possible events that can happen within the simulated environment, allowing UAV to navigate without collisions within simulated environments

Future researches may include adaptation into real world environment on a real drone, testing on environments with dynamically moving obstacles, and attachment of multiple sensor for wider field of view allowing detection of collision from multiple angles. In addition, to accommodate complex environments and larger dataset, a use of incremental learning algorithm could effectively handle limited memory capacity. Also, it is expected that our system combined with effective decision making system to be used at the moment of collision detection, such as CNN for effectively choosing travelling direction, will make the system applicable to path finding in complex environments. Such systems will provide UAVs a far more cost efficient autonomous flight system compared to other systems that only depend on computation heavy data such as images, which also requires large training dataset beforehand.

TABLE I
HIGHEST ACCURACY (%) WITHIN 50 ITERATIONS

Event type	Model trial number		
	1	2	3
Precision (iter #)	100% (36)	100% (46)	100% (39)
Recall (iter #)	98.88% (31)	99.96% (43)	99.04% (46)
Accuracy (iter #)	99.86% (36)	99.99% (44)	99.90% (46)

Acknowledgment

This work is supported by the National Research Foundation of Korea through South Korean Government under Grant NRF- 2016M1B3A1A01937575

References

[1] Y. Lecun and Y. Bengio, “Convolutional Networks for Images, Speech, and Time-Series,” Handb. brain theory neural networks, 1995.

[2] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, “Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness,” 2017.

[3] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, “CNN-based single image obstacle avoidance on a quadrotor,” Proc. - IEEE Int. Conf. Robot. Autom., 2017.

[4] A. Guisti J. Guzzi, D. C. Ciresan, F.-L. He, J.P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al ., “A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots,” Robot. Autom. Lett., vol. 1, no. 2, pp. 661–667, 2016.

[5] D. Gandhi, L. Pinto, and A. Gupta, “Learning to Fly by Crashing,” arXiv:1704.05588, 2017.

[6] F. Sadeghi and S. Levine, “CAD2RL: Real Single-Image Flight without a Single Real Image,” arXiv:1611.04201, 2016.

[7] V. Mnih, Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. Playing Atari with deep reinforcement learning. Technical report. Deepmind Technologies, arXiv:1312.5602, 2013

[8] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation,”arXiv:1703.00420, 2017.

[9] L. Tai and M. Liu, “Towards Cognitive Exploration through Deep Reinforcement Learning for Mobile Robots,” arXiv:1610.01733, 2016.

[10] S. Hochreiter and J. Urgen Schmidhuber, “Long Short-Term Memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[11] D. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization. arXiv: 1412.6980, 2014