aurum

AURUM MÜHENDİSLİK SİSTEMLERİ VE MİMARLIK DERGİSİ
AURUM JOURNAL OF ENGINEERING SYSTEMS AND ARCHITECTURE

Cilt 4, Sayı 2 | Kış 2020
Volume 4, No 2 | Winter 2020, 225-236

ARAŞTIRMA MAKALESİ / RESEARCH ARTICLE

NAVIGATING ROBOTS IN A COMPLEX ENVIRONMENT WITH MOVING OBJECTS USING ARTIFICIAL INTELLIGENCE

Omar Mahmood YASEEN[1]
Altinbas University, Graduate School of Science and Engineering, Electrical and Computer Engineering, Istanbul. omaralsaher1987777@gmail.com ORCID No: 0000-0003-3641-8655

Osman N. UÇAN[2]
Altinbas University, Faculty of Engineering and Natural Sciences, Department of Electrical-Electronics Engineering, Istanbul. osman.ucan@altinbas.edu.tr ORCID No: 0000-0002-4100-0045

Oğuz BAYAT
Altinbas University, Faculty of Engineering and Natural Sciences, Department of Software Engineering, Istanbul. oguz.bayat@altinbas.edu.tr ORCID No: 0000-0001-5988-8882

GELİŞ TARİHİ/RECEIVED DATE: 06.07.2020 KABUL TARİHİ/ACCEPTED DATE: 25.12.2020

Abstract

Robots are being used to automate several tasks in different environments. Some of these applications require the robots to be able to navigate in complex environments and avoid obstacles to reach their destinations. According to the dynamic nature of these environments, Artificial Intelligence (AI) is being used to allow robots handle continuously-changing environments. The existing techniques require intensive processing power and energy sources, which limits their employment is many applications. Thus, a new method is proposed in this study to take control of the robot when a collision is predicted. Different representations of the environment are used, so that, historical information can be provided efficiently. However, the results show that the use of the entire batch has better performance with similar complexity. The proposed method has been able to reduce the number of collision and increasing the speed of the robot during the navigation.

Keywords: Robotics; Artificial Neural Networks; Reinforcement Learning; Light Detection and Ranging.

YAPAY ZEKA KULLANARAK KARMAŞIK BİR ORTAMDA ROBOTLARI HAREKET ETTİRME

Özet

Robotlar, farklı ortamlardaki çeşitli görevleri otomatikleştirmek için kullanılıyor. Bu uygulamalardan bazıları, robotların karmaşık ortamlarda gezinmesini ve hedeflerine ulaşmak için engellerden kaçınmasını gerektirir. Bu ortamların dinamik doğasına göre, robotların sürekli değişen ortamları işlemesine izin vermek için Yapay Zeka (AI) kullanılmaktadır. Mevcut teknikler yoğun işleme gücü ve enerji kaynakları gerektirir, bu da istihdamlarını sınırlayan birçok uygulamadır. Bu nedenle, bu çalışmada bir çarpışma tahmin edildiğinde robotun kontrolünü ele almak için yeni bir yöntem önerilmiştir. Çevrenin farklı gösterimleri kullanılır, böylece tarihsel bilgi verimli bir şekilde sağlanabilir. Ancak sonuçlar, tüm partinin kullanımının benzer karmaşıklıkla daha iyi performansa sahip olduğunu göstermektedir. Önerilen yöntem, navigasyon sırasında çarpışma sayısını azaltabilir ve robotun hızını artırabilir.

Anahtar Kelimeler: Robotik; Yapay Sinir Ağları; Takviye Öğrenimi; Işık Tespiti ve Değişimi.

1. Introduction

With the rapid growth of using robots to automate several tasks in different environments, the challenge of autonomously navigating through the environment has emerged as a limitation toward using these robots. In such an environment, the path from one position to another may not be a straight line, according to the existence of obstacles in the environment. Hence, robots operating in these environments are required to have the ability to navigate throughout these obstacles to reach their destinations. However, according to the dynamic nature of these environments and the possible change in the positions of the obstacles, hard-coding navigation rules can also impose a limitation to the applications that employ these robots (Bottou, 2014; Robert, 2014).

Artificial Intelligence (AI) allows computers to learn directly from the environment, by using examples collected from that environment or by directly interacting with the environment. One of the approaches that are widely used to allow computer-based devices to interact with an environment is Reinforcement Learning (RL). This approach evaluates the state of the agent in the environment, by collecting data through sensors, and executes an action that is optimal to achieve the required task while being in that state. To select the optimal action, the agent is required to predict the outcome of each action, so that, the action with the best outcome is selected for execution. However, predicting such outcomes requires knowledge about the environment, i.e. as an approximation of the function that represents the environment (Mnih et al., 2015; Silver et al., 2018).

Artificial Neural Networks (ANNs) have shown the best performance in approximating complex computations, which has encouraged the use of these networks in RL applications. These networks require training to be able to approximate the function that represents the environment, i.e. the outcome of each action at a certain state. For this purpose, the feedback collected from the environment, known as the reward value, is used to train the neural network, so that, the neural network gains the ability to predict the reward value of each possible action at a certain state. These predictions are used to direct the agent to select the action with the highest reward value, i.e. predict the feedback of the environment before executing the action, by selecting the action that is predicted to return the highest reward value (Lillicrap et al., 2015).

As the ANNs are used in RL to predict the reward value, denoted as Q, and according to the use of deep neural networks, which contain more than one hidden layer, such network is known as DQN. The use of DQN to navigate robots in an environment has been investigated in several studies. These studies have shown that by making historical data about the movement of the agent has improved the navigation capabilities of the robot. However, the approaches used to feed historical data back to the neural network has dramatically increased the complexity of the DQN, which requires intensive processing. Moreover, these methods do not consider the movement of other objects in the environment. Such movement can produce unexpected behavior by the agent, especially with the existence of historical data, as the agent expects the obstacle at a certain position but the sensors return different data (Choi, Park, Kim, & Seok, 2019; Kahn, Villaflor, Pong, Abbeel, & Levine, 2017; Kim, Kim, & Lee, 2018).

In this study, a novel DQN-based method is proposed to allow robots navigate in a complex environment with moving objects. The proposed method uses a simpler approach to feed data about the environment

to the DQN. Hence, less processing power is required to select the actions by the agent, which reduces the response time of the agent. Moreover, the proposed method considers the movement of other objects in the environment, so that, the agent can predict the trajectory of these object and avoid colliding with them.

2. Literature Review

Reinforcement learning uses the concepts of agents, environments, states, actions and rewards (Ha & Schmidhuber, 2018; Littman, 1994; Tan, 1993; Watkins & Dayan, 1992). As shown in Figure 2.1, the agent selects an action to be executed in the environment, which returns the agent's new state in addition to its new state. This procedure is repeated until the agent reaches the required state, i.e. finishes the execution of the required task. However, the agent does not have any knowledge about the environment and how the new state and reward are returns for a certain action at a certain state. Thus, in reinforcement learning, the agent attempts to predict the actions that maximize the rewards received from the environment, by approximating a function that represents the environment and how it responds to the actions (Mnih et al., 2013).

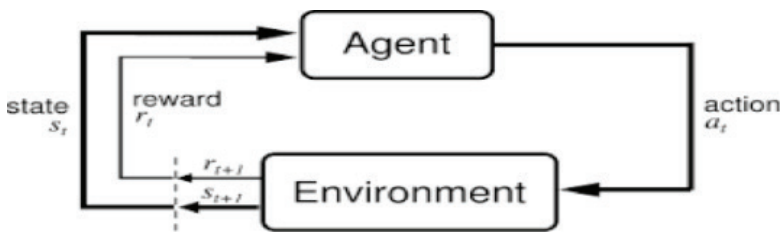

Figure 1: Illustration of the interaction between the Agent and the Environment in reinforcement learning.

DQNs are used to approximate these functions and have been able to significantly improve the performance agents that use RL. This approach has been used in previous studies to enable robots to navigate in complex environments. The method proposed by Kim et al. (Kim et al., 2018) uses a DQN to navigate robots in an environment with obstacles. This method uses a Light Detection and Ranging (LiDAR) sensor to measure the distance between the robot and its surrounding. A memory batch is used to provide the DQN with historical data about the state and actions of the agent before the current state. The enormous amount of data in this approach requires high computing resources from the robot, which requires expensive equipment and high-power energy sources.

Another method is proposed by Choi et al. (Choi et al., 2019), which aims to use less-expensive equipment, compared to the LiDAR sensor. This method uses a depth camera to investigate the surroundings of the robot. However, the DQN used in this approach uses the Long- Short-Term Memory approach, which allows the neural network to consider earlier decisions in the current one. Despite the reduction in the cost by eliminating the use of the LiDAR, this method also requires intensive processing, which in this case poses a limitation to the employment of the method. Moreover, the accuracy of the measurements collected by the depth camera is significantly lower than that when the LiDAR is used.

3. Methodology

According to the better accuracy of the LiDAR sensor, it is used in the proposed method to measure the distance between the robot and its surroundings. The neural network used as the DQN of the proposed method is a Convolutional Neural Network (CNN), which is significantly less complex than the LSTM in terms of computations complexity and is capable of handling three-dimensional inputs. The need for three-dimensional input is to allow historical representation of the environment surrounding the robot. Such representation allows the robot to distinguish its own path and the paths of the objects moving in the environment, with respect to its movement.

3.1. Environment Representation

The data collected from the LiDAR represents the distance to the nearest object at each angle, with 1º resolution to cover the entire 360º radius. Accordingly, each of the figures shown in Figure 2 can be created based on the measured distances. Moreover, as shown in Figure 2, the movement of the object marked in red can be concluded, as the remaining object are still in the generated output. Thus, by providing such historical data, the environment can be described to the reinforcement learning agent, which allows making the appropriate decisions.

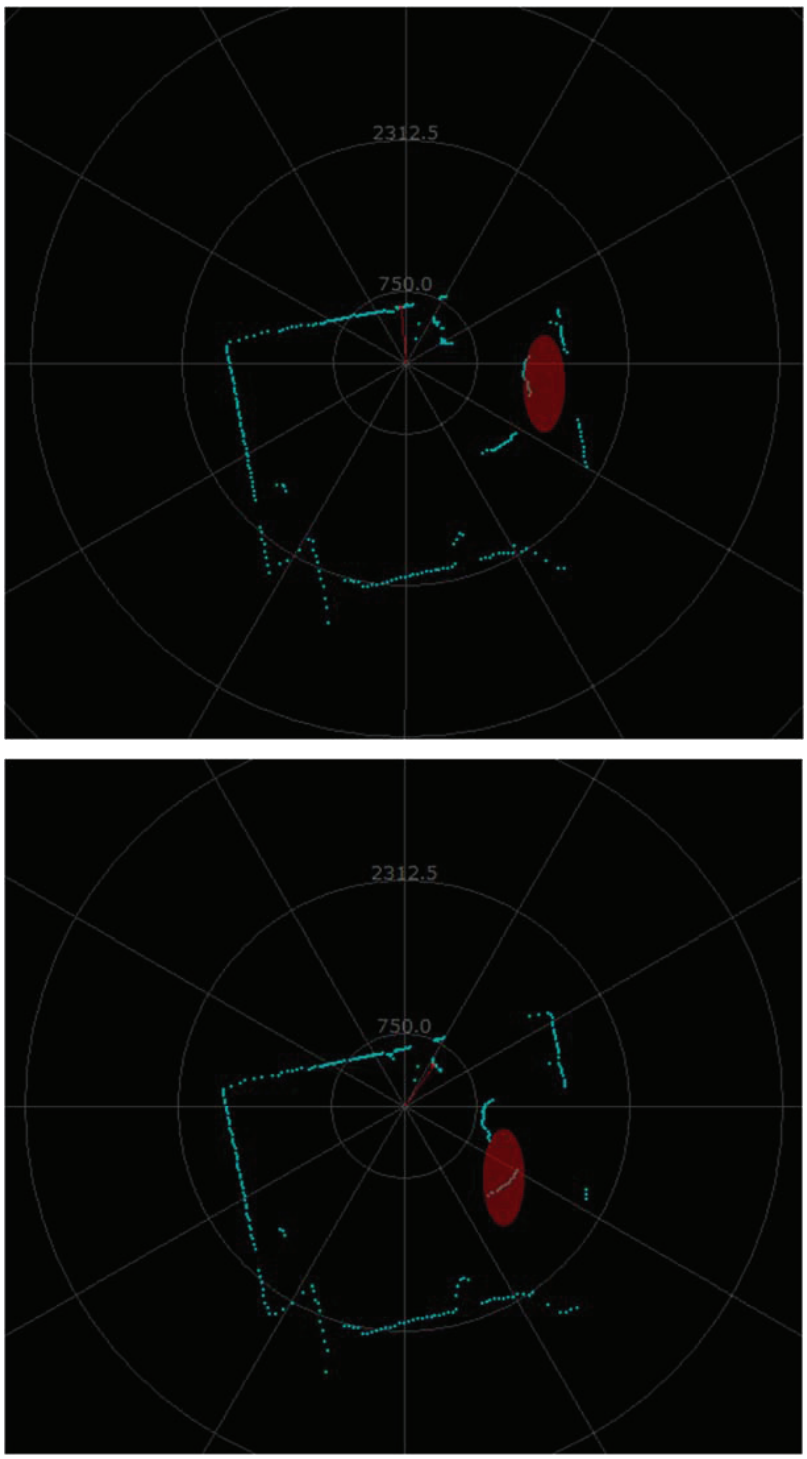

Figure 2: Representation of the outputs collected from the LiDAR. Top: Previous scan; Bottom: Current Scan.

With a scanning frequency of 10Hz for the LiDAR, movement of slower objects can be difficult to detect, especially in complex environments. Hence, two approaches are used in the proposed method. The first approach collects data of 10 scans from the LiDAR and provide it to the agent. This allows representing the entire last second to the agent with all the objects in the environment. Despite this accurate representation, processing such an amount of data requires complex models, as well as extensive processing power. Thus, in the second approach the entire second is represented using three scans only. To represent the environment at time instance *T*, the first scan is collected at *T-1sec*, while the second and third scans are collected at *T-0.5sec* and *T*. Despite the less information provided to the agent, this approach can be considered significantly more efficient, if and only if it can provide similar performance to the use of all the scans.

3.2. Agent Model

As the proposed method is design to interevent with the existing navigation system and override the command from that system when a collision is predicted and navigation is overridden, the commands being sent to the robot to navigate must pass through the agent. Hence, these commands must be inputted to the agent, in addition to the environment representation, which requires a hybrid neural network to handle the different types of inputs. For different navigation systems, with *C* number of commands, and *L* scans collected using the two different approaches proposed in this study, the agent in the proposed method uses the neural network shown in Figure 3.

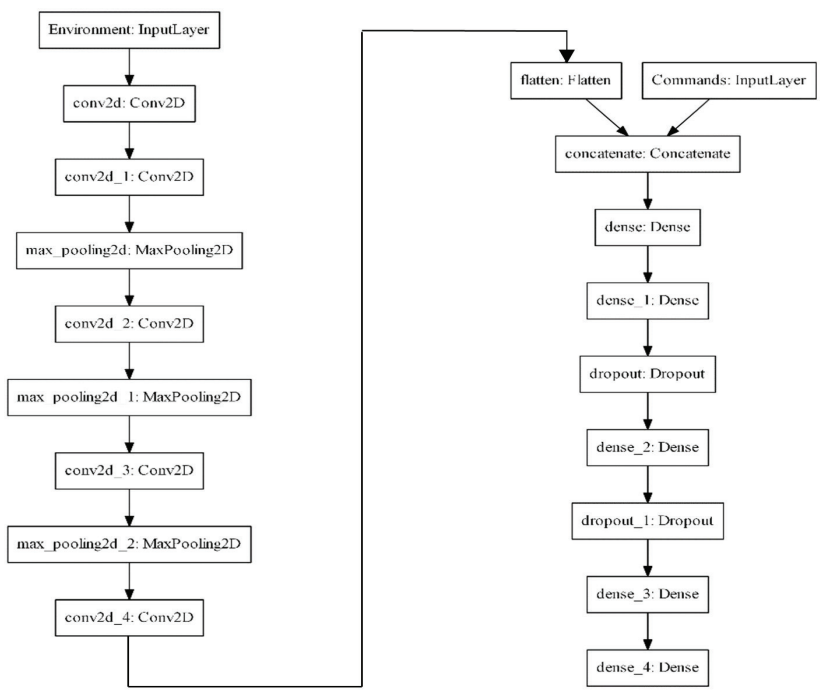

Figure 3: Structure of the agent's neural network.

As shown in Figure 3, the multi-dimensional input that represents the environment is first processed using a set of convolutional layers before being flattened and concatenated to the inputs collected from the commands required to be delivered to the robot. According, the proposed method can forward these commands as long as the environment seems to be safe for such a command to be executed. Alternatively, the proposed method can take control and override the navigation commands when the execution of such commands is predicted to cause a collision, including being stationary when an object is moving towards the robot. Thus, the number of neurons in the output layer is equal to the number of neurons in the "Commands" input layer, so that, the proposed method is transparent to the currently working system. Additionally, the size of the input layer "Environment" is set to 100×100×L, where *L* is the number of scans collected form the LiDAR.

3.3. Training the Agent

Training reinforcement learning agents relies on the values of the rewards calculated for the selected actions. Hence, these reward values are required to reflect the behavior required from the agent. In the proposed method, the agent is required to follow the commands incoming from the navigation system, unless a collision is expected to occur based on the state of the agent in the environment. Thus, a reward value of zero is provided to the agent, so that, the values that are used to train the agent are equal to the commands incoming from the navigation system, as long as there are no collisions. When a collision occurs, a negative value of -1 is provided, so that, the agent avoids this command, or action, at that state. An object is considered collided with the agent when its distance from the robot is less than 15cm, which is the minimum range of the LiDAR.

4. Experiments and Discussion

In order to evaluate the performance of the proposed method, a robotic vehicle is implemented as shown in Figure 4. A Raspberry Pi Zero-w, which has a built-in wireless module for WiFi and Bluetooth connections, is used to implement the proposed reinforcement learning method. Controls are sent from a webpage hosted on the controller, which allows access by any remote navigation system. For evaluation purposes, a manual navigation system is used by sending the commands manually through the webpage. The proposed method is implemented using Python Programming Language with the Tensorflow deep learning library to implement the proposed neural network for the agent.

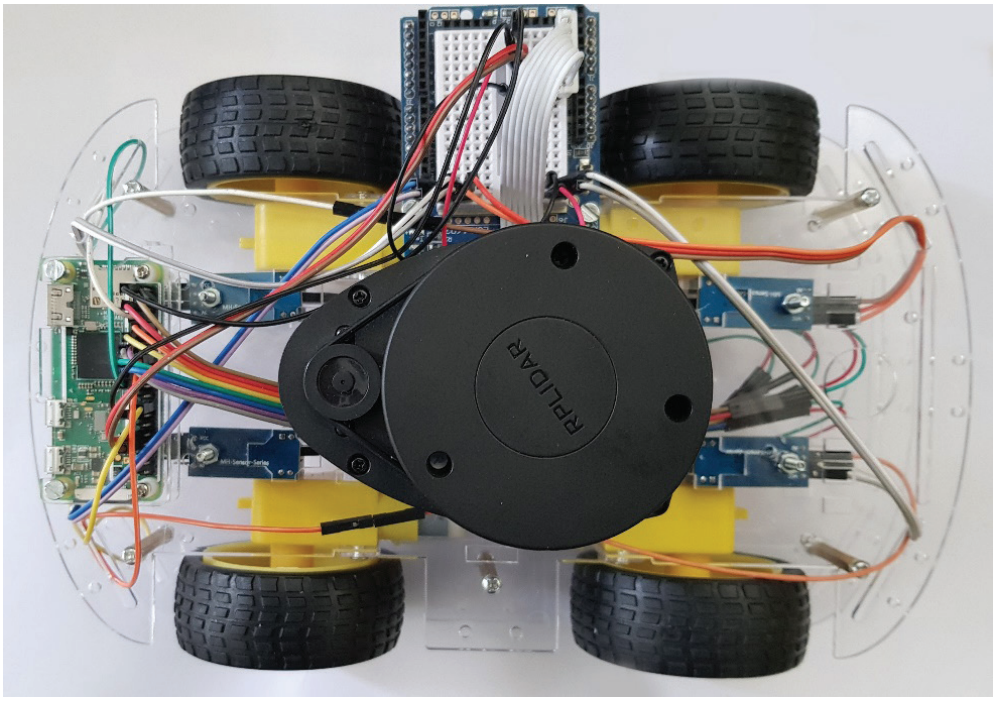

Figure 4: Illustration of the implemented robotic vehicle.

Similar to the training procedure conducted by Kahn et al. (Kahn et al., 2017), the agent is trained for 10 iterations using the proposed approach, where five rollouts are performed per each iteration, with four different initial positions, i.e. states. An iteration is considered finished when the robot collides with an object, moving or stationary, or when a maximum of 10 time steps elapse. Hence, the robot maintains movement for an average of fifteen minutes per each iteration. As shown in Figure 5, which shows the number of collisions occur at each speed level and above, the proposed method has been able to significantly reduce the number of collisions at all speed levels, especially at lower speeds. This reduction is a result of providing the robot with the ability to move from stationary to allow other moving objects in the environment to cross. Moreover, the ability to predict the trajectory of these objects allows the robot to deviate from collisions.

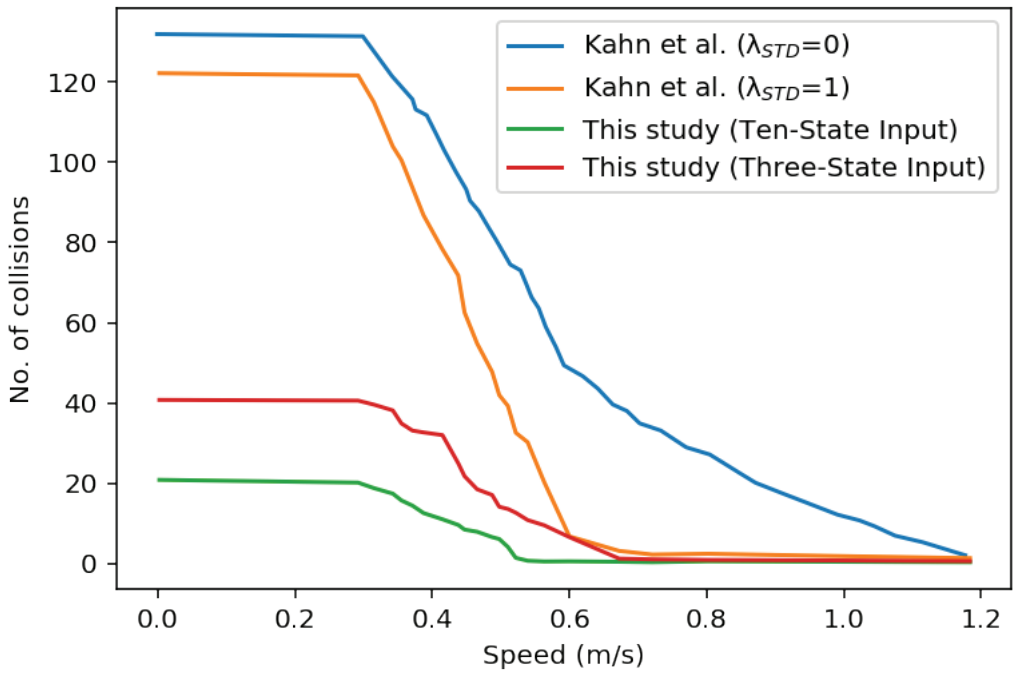

Figure 5: Number of collisions versus speed.

Figure 5 also shows that the use of 10 LiDAR scans per each input to the agent has significantly better performance than only using 3 scans. Moreover, despite the lower complexity of the neural network required to process the three-scan inputs, the time required to process these inputs has been very similar to the use of ten-scan inputs. This similarity is according to the additional computations required to store the previous scans and collect them when required by each input. Thus, the use of the proposed method has been able to produce significantly better navigations for the robot, by avoiding collisions with other objects in the environment. Moreover, the use of ten-scan input is also more efficient, according to the better performance and similar overall complexity.

In addition to the lower number of collisions occurred when using the proposed method, the average speed of the vehicle has also been better, as shown in Figure 6. This improvement is according to the ability of the proposed method to change direction while maintaining speed rather than reducing speed to avoid to collision. Such behavior is achieved by enforcing the commands incoming from the controller as long as a collision is not predicted to happen. Moreover, the use of the three-scan inputs has not been able to maintain the speed of the robot during the training, according to the missing information in the representation.

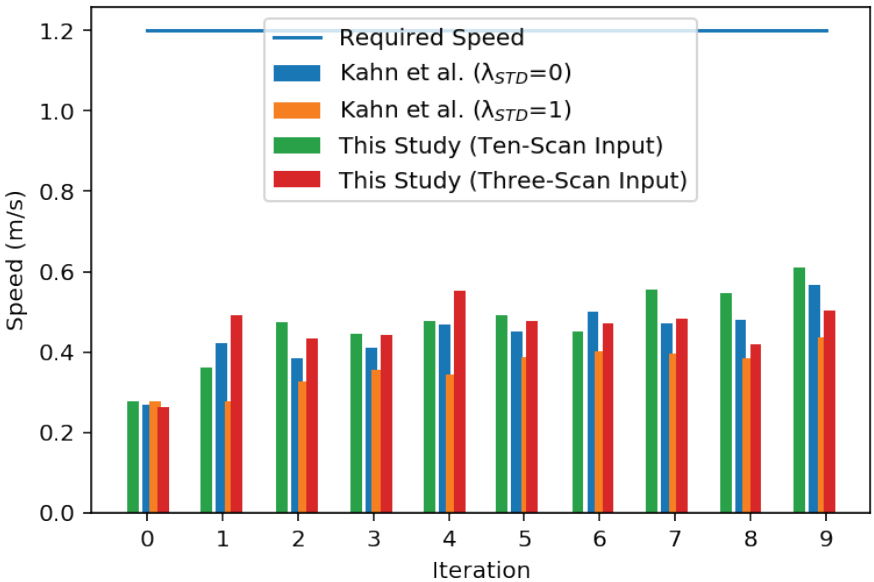

Figure 6: Speed of the robot in the environment versus training epochs.

The results shown in Figure 7 also show that the use of the proposed method has been able to achieve faster learning and more accurate decisions to avoid collisions. This learning is illustrated by the percentage of the successful drives that are completed without collision, despite the existence of other moving objects in the environment.

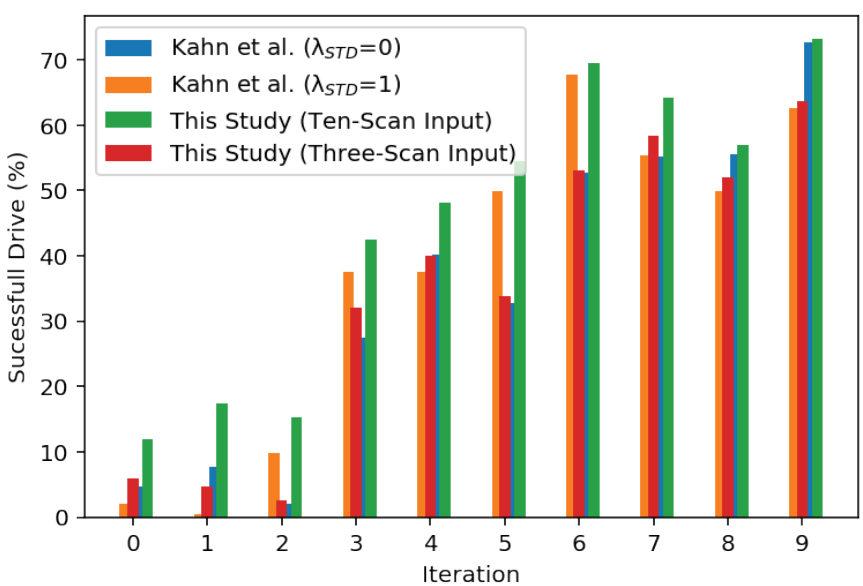

Figure 7: Percentage of successful drives versus iterations.

5. Conclusion

A new collision avoidance method is proposed in this study to improve the navigation of robots in complex environment. The proposed method acts as an intermediate stage between the navigator and the parts of the robot responsible for achieving the movement. When a collision if not predicted to occur, the output of the proposed method is identical to the output collected from the controller. Otherwise, i.e. if a collision is predicted to occur, the proposed method takes control of the robot and override any commands incoming from the navigator. The commands outputted from the proposed method in such case is to avoid the collision regardless of the reason behind the collision.

A deep reinforcement learning approach is used for the proposed method in which the agent estimates the state of the agent in the environment based on the representation of the surroundings, i.e. the distance between the robot and nearest object. These measurements are collected using a LiDAR and forwarded to the agent. In order to allow the prediction of the trajectory of the moving objects in the environment, historical data are provided to the agent, i.e. previous scans from the LiDAR. Two approaches are used to represent such historical data, in which an approach samples the last second by collecting three samples only. Despite the expectations to increase the efficiency of the model by reducing the data, storing and retrieving the required scans has increased the complexity of the model, so that, it became similar to the complexity of the model required to process the entire input batch, i.e. of ten scans.

In future work, servo motors are going to be used to tilt the LiDAR sensor on both the x and y axes, so that, a 3D overview of the environment can be produced to the DQN. The angle of the sensor is also going to be determined by the RL model, according to its navigation requirements. Despite the possible improvement in the performance of the agent, according to the better overview collected from the environment, a more complex DQN is required to control the orientation of the LiDAR. Such increased complexity requires more resources as well as more training for the neural network to detect the relation between the angle of the sensor and the data collected for navigation.

6. References

Bottou, L. 2014. From machine learning to machine reasoning. *Machine learning, 94*(2), 133-149.

Choi, J., Park, K., Kim, M., & Seok, S. 2019. *Deep Reinforcement Learning of Navigation in a Complex and Crowded Environment with a Limited Field of View.* Paper presented at the 2019 International Conference on Robotics and Automation (ICRA).

Ha, D., & Schmidhuber, J. 2018. *Recurrent world models facilitate policy evolution.* Paper presented at the Advances in Neural Information Processing Systems.

Kahn, G., Villaflor, A., Pong, V., Abbeel, P., & Levine, S. 2017. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*.

Kim, K.-S., Kim, D.-E., & Lee, J.-M. 2018. *Deep Learning Based on Smooth Driving for Autonomous Navigation.* Paper presented at the 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., . . . Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning *Machine learning proceedings 1994* (pp. 157-163): Elsevier.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Ostrovski, G. 2015. Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.

Robert, C. 2014. Machine learning, a probabilistic perspective: Taylor & Francis.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., . . . Graepel, T. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140-1144.

Tan, M. 1993. *Multi-agent reinforcement learning: Independent vs. cooperative agents.* Paper presented at the Proceedings of the tenth international conference on machine learning.

Watkins, C. J., & Dayan, P. 1992. Q-learning. *Machine learning, 8*(3-4), 279-292.