Model-Based Methods for Autonomous Follow-Ahead with Obstacle Avoidance

Anthony Cheung[1], Emma Hughson[1], and Khizr Ali Pardhan[2]
https://github.com/alik604/ra

***Abstract*— This study expands on a solution to solve the follow-ahead application where the robot attempts to stay ahead of a human as a human walks around. However, the previous solution did not consider obstacles or model-based methods. As such, the current study will implement 3 methods: (1) a novel model-based method, (2) a traditional model-based method, and (3) a solution without machine learning. The novel model-based method used human intent to find a path for the robot. The traditional model-based method was an implementation of World Model and used human and robot relative states. Finally, the approach without machine learning incorporated a distance heuristic to generate a goal for the robot. For obstacle avoidance and path finding we utilized a trajectory planner to avoid obstacles. In contrast to the original follow-ahead solution we developed methods that required little or no learning at all while solving the obstacle avoidance problem.**

I. INTRODUCTION

Autonomous technologies have proven to ease everyday lives. Specifically, there has been a rise in the need for follow-behind technologies, because it can remove burden and allow human users to expand their potential for multi-tasking. Examples of such technologies are grocery carts that assist in grocery shopping or luggage following behind users in an airport [1, 2]. However, there is one common issue that is ignored with follow-behind robots which is the lack of security [3]. Since users cannot see their luggage when walking around an airport it leaves their valuables open to be easily stolen. As such, we propose using model-based methods to allow a robot to follow-ahead of a human to provide the security that is absent in follow-behind technologies. The current study will compare model-based methods against a baseline that does not require any machine learning.

II. RELATED WORKS

A. Follow-Ahead Motivation

As previously mentioned, follow-behind technologies are commonly implemented in the world of autonomous human-robot interaction. However, there has been an increase in follow-ahead implementations. For instance, Nikdel et al. [4] combined deep learning (i.e., Deep Deterministic Policy Gradient (D4PG)) and trajectory planning to implement follow-ahead. Using D4PG, they were able to determine a short-term goal (i.e., be in-front of human) that was used for the trajectory planner. This combination of D4PG and trajectory planning outperformed current methods that use EKF to predict the human target. However, they assumed an obstacle free environment and used a model-free method. As such, the current study will be considering obstacles in the environment.

B. Model-Based Methods

Model-free methods have been a popular way to implement reinforcement learning. However, model-based methods, although less explored, have shown that they can be better performing, increased data-efficiency, and can take less computation time [5]. Model-based methods are given a model of the world and use this model to predict next states and rewards [6, 7]. One study by Brunnbauer et al. [6] compared model-based and model-free methods for autonomous racing cars using occupancy maps. The model-based method is called Dreamer (i.e., World Model algorithm) and it has shown increased performance over leading model-free methods. For the model-free method they used a range of leading algorithms, such as D4PG and Proximal Policy Optimization (PPO). The model-based method outperformed the model-free methods significantly. In fact, the model-free methods could not complete a single lap in complex maps, whereas the model-based method successfully learned the maps regardless of complexity.

C. Obstacle Avoidance

Currently, obstacles are a constraint in human-robot interaction that has created limitations when deploying both follow-behind and follow-ahead technologies. Obstacle avoidance is a challenging problem as current algorithms are not robust in dynamic and unknown environments [8]. For example, Gockley et al. [9] noted two algorithms for follow-behind robots. The first algorithm is called direction follow, which has robots follow in the general direction of a human subject. The second algorithm is called path following where the robot follows and anticipates the exact path of the human. Participants found when interacting with a robot that was using the second algorithm for navigation it had more human-like behaviour. However, this human-like behaviour went away when the robot had to also avoid obstacles. It is increasingly important to consider obstacles as it can increase the safety of the human we are using or interacting with the robots.

III. METHODOLOGY

The goal of the current paper was to build a model-based solution and add obstacle avoidance. Three approaches were

[1]School of Computing Science, Simon Fraser University, Canada. akc29@sfu.ca, ehughson@sfu.ca
[2]Cognitive Science Program, Simon Fraser University, Canada. kpardhan@sfu.ca

taken: (1) a traditional model-based algorithm called World Model, (2) our own model-based solution, using a Human Intent Neural Network (HINN), and (3) a Trajectory Planner solution used as a baseline. All three implementations use the human's feature space as input. In comparison with other work, our work investigates model-based methods in conjunction with obstacle avoidance.

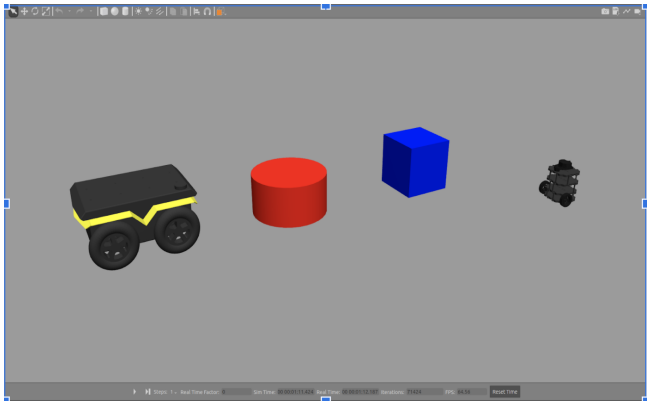

Fig. 1. Gazebo environment showing the Jackal robot (left), obstacles (center), and Turtlebot 3 burger robot (right).

A. Environment Software Stack Description

The environment setup was implemented using ROS [10], Gazebo[11], and AI Gym [12]. ROS was used to communicate with Gazebo and to control the robots. Gazebo was used to simulate our environment with robots and obstacles(As seen on the left). The ROS trajectory planner, specifically the TEB Local planner, was used to avoid obstacles for both robots. The human robot was a Turtlebot burger robot (small black robot), the robot was a Jackal Robot (large yellow and black robot). Dynamic obstacles were cylinders and cubes, as shown in Figure 1. For brevity, we will be referring to the human robot as the human and the jackal robot as the robot hereafter.

B. Environment Initialization

For testing purposes, the robots were placed randomly around the center of the environment and obstacles were randomly placed around the robot. Obstacle positions were taken directly from a Gazebo output and sent to the trajectory planner to allow obstacle avoidance.

To increase generalizability of model learning, we implemented 5 different paths with different complexities that the human would take. The first path was the simplest human trajectory, a straight-line. The second path and third path included either a left curve or a right curve. The fourth path involved both left and right curves to create a zig-zag pattern. The final path was a random path, where the human would randomly choose a different heading every 5 seconds. Rewards were calculated using the same reward calculation described in Nikdel et al. [4, Fig. 3]. A positive reward is given for every time step the robot is ahead of the human. Negative rewards are given if the robot is not ahead of the human. For evaluation, the stochasticity in environment initialization was removed to evaluate each pipeline. Robots and obstacles were placed and oriented based on a pattern allowing direct comparison of evaluation results. The random human path was also deactivated.

C. Trajectory Planner Solution

The trajectory planner solution involves no machine learning. This solution uses the ROS navigation stack and more specifically the trajectory planner to navigate and move the robot ahead of the human. Using a distance heuristic we generate a goal and give it to the trajectory planner to plan an optimal trajectory for our robot to take. The distance heuristic takes a vector containing coordinates that describe the target distance (i.e., being 0.4 meters ahead of human), as well as the human state (x^h, y^h, θ^h). The coordinates are then translated using a rotation function that translates the vector using θ^h, $(x^{new\_target}, y^{new\_target})$, then the goal is calculated by adding the vector to the human position (x^h, y^h). Goal is defined as:

$$\begin{bmatrix} x^{new\_target} \\ y^{new\_target} \end{bmatrix} = \begin{bmatrix} cos(\theta^h) * x^{target} - sin(\theta^h) * y^{target} \\ sin(\theta^h) * x^{target} + cos(\theta^h) * y^{target} \end{bmatrix}$$

$$Goal = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x^h + x^{new\_target} \\ y^h + y^{new\_target} \end{bmatrix}$$

D. Human Intent Neural Network

The HINN is a feedforward neural network that takes in $(\dot{x}^h, \dot{y}^h, \theta^h, \upsilon^h, \omega^h)$ describing the human state and the area around the human. To describe the human state, we included the human's velocity, heading, and last 5 positions in history. A reduced laser scan was used to describe the area around the human to allow the HINN to "see" obstacles. Data was collected by cycling through different human trajectories, while randomly positioning obstacles around the human. The output of the HINN is a vector, $(x^{h'}, y^{h'}, \theta^{h'})$, which is a human intent prediction.

Two algorithms were implemented to select the optimal goal for the robot to go to: (1) Distance Heuristic, and (2) Monte Carlo Tree Search (MCTS). The distance heuristic is the same as described in the previous section. However, instead of inputting human state we input the HINN's human intent prediction to calculate the goal. The MCTS algorithm also uses heuristics to select paths recursively in an asymmetric tree. Once it reaches a frontier, it expands the next node, or action. MCTS then runs a simulation of the current problem, in our case a trajectory. As the simulation occurs information is collected about the optimal location (i.e., ahead of a human) for the robot to be given the input from the HINN. All nodes on a given path are updated with the collected information. Simulating future actions allows MCTS to pick the optimal trajectory for the robot to take by choosing the best path in the tree [13].

E. World Model

Traditional World Model consists of three main components: (1) Variational Autoencoder (VAE), (2) Long Short Term Memory (LSTM), and (3) a controller [15]. The VAE uses convolutional layers to extract data at the pixel level environment and encode it into a smaller dimensional feature space. However, because we are directly getting input from ROS regarding the position of human and robot, we removed the VAE from our implementation of World Models. The LSTM takes in an observation, which is a vector that contains information about the robot (z_t^r)and human's relative state (z_{t-1}^h) history as defined in Nikdel et al. [4, eq. (1)][4, eq. (2)]. After training the LSTM, the Controller is then trained. The controller uses a Covariance Matrix Adaptation Evolution Strategy which is a derivative-free optimization strategy [14]. In our implementation of World Models, the Controller takes as input the hidden state from the LSTM and the robot state relative to the human state previously defined. The World Model then gives ROS the next action for the robot to take.

IV. EXPERIMENTAL RESULTS

The results for the Trajectory Planner solution and the HINN solution are shown in Figure 2. As we can see from the graph, both solutions performed fairly well while avoiding obstacles. A consistent increase in reward was seen as episodes increased. However, the trajectory planner solution clearly outperforms the HINN solution. This makes sense because the HINN solution uses a prediction to solve for a goal instead of using the exact human position. As the episodes increased there was a steady increase in separation between the two lines.

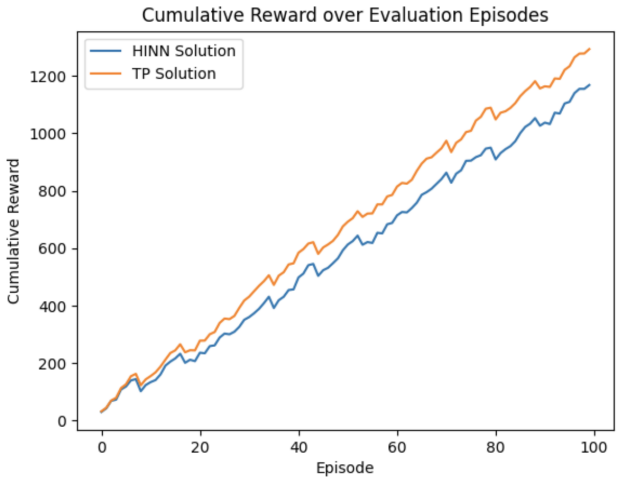

Fig. 2. The cumulative rewards of the HINN and Trajectory Planner solution over 100 episodes.

For the World Model, there was no apparent learning as the episodes increased. More specifically, from Figure 3, we can see a steady decrease in cumulative rewards for the LSTM as time went on. This could either demonstrate the LSTM's goal not to learn but rather to give a high quality hidden units to the controller. Furthermore, the controller, as shown in Figure 4, started out low and steadily increased and plateaued at an cumulative reward of 65. For comparison to the World Model (a model-based method) the performance of the model-free method, Twin Delayed Deep Deterministic Policy Gradient (TD-DDPG), as shown in Figure 5 and demonstrates a slow and erratic gain in rewards over 16000 episodes. This model was given the same input as the World Model. Although the number of episodes is significantly different, we can see that the World Model had a similar performance outcome at the 2000 episode mark comparatively using a moving average.

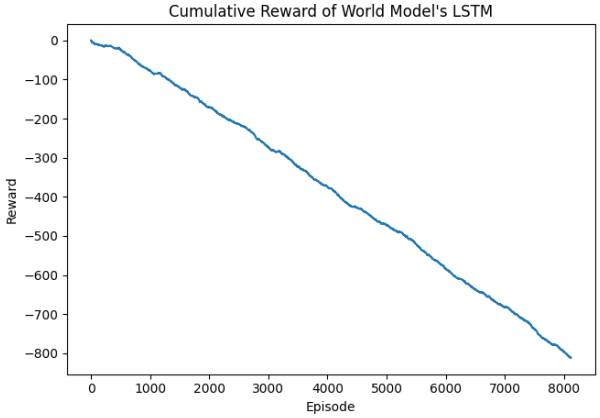

Fig. 3. The cumulative reward of the World Model's LSTM over 8000 episodes.

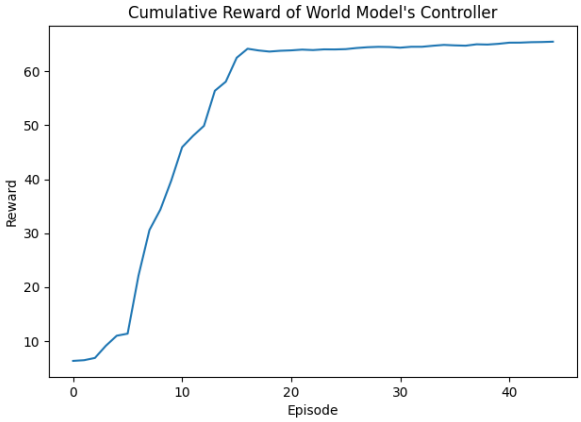

Fig. 4. The cumulative reward of the World Model's controller over 40 episodes.

V. DISCUSSION

Fundamentally, the current study aimed to investigate if model-based methods proved useful in solving the follow-ahead problem. Although the World Model implementation had decreasing rewards, HINN showed that they could gain rewards demonstrating some learning. However, the Trajectory Planner solution provided the best outcomes. Not only did the Trajectory outperform the HINN, it also regularly

Fig. 5. The running average over 16000 episodes on the TD-DDPG.

reached its goal more often than the HINN. Thus providing evidence that using machine learning may not be the best method for the follow-ahead problem.

As mentioned, one of the problems we solved was obstacle avoidance. Both the HINN and trajectory planner solution were good at avoiding obstacles. This was largely part of the ROS navigation stack was very successful when finding optimal trajectories. We also considered using a Radical Basis Function (RBF), which has function approximation abilities, in place of the feedforward neural network but the literature lacked insight into how RBF would perform in human intent prediction.

The World Model overall performed poorly. The rewards for the LSTM had a steady decline while the rewards for the controller settled and converged at 65. This demonstrates that the World Model did not appear to learn overtime and got stuck at a local optimum. This could be due to the eradication of the VAE, which is an essential part of World Model implementation. Another possible reason for the performance decrease is that World Model is normally performed on significantly different problem spaces, such as Brunnbauer et al. [6], who used occupancy maps to predict racing car paths. Furthermore, it could be also do to the lack of available parameters. For example the controller was only a single dense layer while the LSTM was a single layer followed by dense layers. We could have made the LSTM stacked, meaning to have multiple LSTM layers, giving us more parameters to learn the system dynamics.

VI. FUTURE WORK

A major limitation was time. Certain ideas had to be scrapped part way through development to ensure core components to be accomplished. As such, the MCTS was not deployed when training the HINN. However, it is implemented and has been prepared on the Lunar Lander gym environment. We hope to fully implement this MCTS algorithm on the HINN in the future as MCTS has proven significantly helpfully in game problem spaces. This also highlighted the difficulties to know when to give up on an algorithm for the sake of completion. In addition, we also hope to implement VAE using a occupancy map of the environment on ROS. This could improve the World Model performance and could demonstrate that the model-based methods are a good approach for the follow-ahead problem.

Another area to make more applicable to the real world is using a laser scan for obstacle avoidance. Currently, the system uses raw positions from Gazebo for the trajectory planner. These raw positions are tough to get in the real world and it would make more sense to use a laser scan and create a cost-map for the trajectory planner. However, we decided not to pursue this path due to lack of time.

Another limitation was that we had only one member who had a significantly good CPU, while another had a good GPU. We did not have time to optimize training and data was collected in a one-by-one manner which leads to significant memory copy overhead costs. Thus, a CPU outperformed a RTX 3060 TI, so we didn't use CUDA. As such, we could only use those two available computers for debugging on the gym environment and training to get results. This created time constraints when developing our models as it required large amounts of time to just train one model. We did not have the means to optimize hyper parameters, particular for model architecture. In the future we would leverage a cloud computing platform or increase the availability of applicable machines.

VII. CONCLUSIONS

The results of our study highlighted that model-based methods were outperformed by a non-machine learning method (i.e., Trajectory Planner Solution). Furthermore, that machine learning is not the end-all-be-all when it comes to implementing follow-ahead tasks. This was demonstrated by the trajectory planner solution using the distance heuristic. We also recognize that future work needs to be done on optimizing the World Model implementation and making the trajectory planner solution more transferable to the real world.

ACKNOWLEDGMENT

We would like to acknowledge work of Scott Harrison and the mentorship of Payam Nikdel the author of the paper who we based our work on.

REFERENCES

[1] V. Kulyukin, C. Gharpure, and J. Nicholson, "RoboCart: toward robot-assisted navigation of grocery stores by the visually impaired," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2845–2850, Sep. 2005. doi:10.1109/IROS.2005.1545107

[2] P. L. S. Krishnan, R. Valli, R. Priya, and V. Pravinkumar, "Smart Luggage Carrier system with Theft Prevention and Real Time TrackingUsing Nano Arduino structure," 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), pp. 1–5, 2020. doi: 10.1109/ICSCAN49426.2020.9262445.

[3] W. Mi, X. Wang, P. Ren and C. Hou, "A system for an anticipative front human following robot.", In Proceedings of the International Conference on Artificial Intelligence and Robotics and the International Conference on Automation, Control and Robotics Engineering (ICAIR-CACRE '16), pp. 1-6, 2016. doi:https://doi.org/10.1145/2952744.2952748.

[4] P. Nikdel, R. Vaughan, and M. Chen, "LBGP: Learning Based Goal Planning for Autonomous Following in Front," ArXiv, 2020.

[5] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," International Conference on Learning Representations, 2020. https://openreview.net/forum?id=S1lOTC4tDS

[6] A. Brunnbauer, L. Berducci, A. Brandstätter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Model-based versus model-free deep reinforcement learning for autonomous racing cars," ArXiv, 2021.
[7] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan , C. Finn, P. Kozakowski, S. Levine , A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski, "Model-Based Reinforcement Learning for Atari," ArXiv, 2020.
[8] E. Kelasidi, S. Moe, K. Y. Pettersen, A. M. Kohl, P. Liljebäck, and J. T. Gravdahl, "Path Following, Obstacle Detection and Obstacle Avoidance for Thrusted Underwater Snake Robots," Frontiers in Robotics and AI, vol. 6, p. 57, Jul. 2019. doi:10.3389/frobt.2019.00057
[9] R. Gockley, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," Proceeding of the ACM/IEEE international conference on Human-robot interaction - HRI '07, pp. 17–24, Mar. 2007. doi:https://doi.org/10.1145/1228716.1228720.
[10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in ICRA workshop on open source software, vol. 3, no. 3.2. Kobe, 2009, p. 5.
[11] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), vol. 3, 2004, pp. 2149–2154 vol.3.
[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," ArXiv, 2016.
[13] T. Vodopivec, S. Samothrakis, and B. Ster, "On Monte Carlo Tree Search and Reinforcement Learning," Journal of Artificial Intelligence Research, vol. 60, pp. 881–936, Dec. 2017.doi:10.1613/jair.5507
[14] V. H. Dang, N. A. Vien, and T. C. Chung, "A covariance matrix adaptation evolution strategy in reproducing kernel Hilbert space," Genetic Programming and Evolvable Machines, vol. 20, no. 4, pp. 479–501, 2019. doi:10.1007/s10710-019-09357-1.
[15] D. Ha and J. Schmidhuber, "Recurrent World Models Facilitate Policy Evolution," ArXiv, 2018.

Anthony Cheung	Emma Hughson	Khizr Ali Pardhan
Project Work: - Software Stack Setup and Debug - Obstacle Generation and Environment Initialization - Robot Navigation and Obstacle Avoidance - Distance Heuristic and Trajectory Planner Solution - HINN Pipeline Infrastructure - Wrote/Edited Report - Wrote/Edited Poster	Project Work: - Implemented Monte Carlo Tree Search - Implemented World Model - Created Poster - Wrote project Milestone #1 - Wrote project Milestone #2 - Wrote Introduction and Related Works in Report - Researched information regarding follow-ahead implementations - Wrote methods, results, discussion, and conclusions	Project Work: - Sample Code on several Model-Less RL - Implemented TD-DDPG to compete against D4PG - Implemented Human Intent Neural Network model - Debugging Monte Carlo Tree Search - Debugging World Models, adapting to our continuous ROS environment, rewrote to single processed - Trained and Evaluated the above RL models - Wrote/Edited The Poster and Report

Fig. 6. Contributions