Follow-Ahead Human-Robot Navigation via Monte Carlo Tree Search and Deep Reinforcement Learning

Archita Srivastava, Ankush Singh, Gemmin Sugiura, Jonathan Ung

***Abstract*—We replicate the MCTS-DRL framework of Leisiazar et al. [1], [2] for robotic follow-ahead navigation. The method integrates Monte Carlo Tree Search with a pretrained RL value function, using learned value estimates in place of random rollouts to generate consistent short-term navigational goals, with an LSTM-based human action predictor biasing tree expansion toward more probable human futures. We re-implement the full framework from scratch in ROS2, including the MCTS planner, RL value function, and LSTM predictor, and deploy the system on a QBot 2e using VICON for pose estimation and Cartographer for mapping. Experiments on a range of trajectories probe follow-ahead behavior under controlled conditions. Project page and videos are available at [our website].**

***Index Terms*—human-robot interaction, follow-ahead, MCTS, deep reinforcement learning, LSTM**

I. Introduction

Human-following robots (HFR) maintain a consistent distance and orientation relative to a target person. Applications include autonomous luggage carts, search and rescue, activity filming, and patient monitoring. HFRs are broadly categorized as follow-behind, side-by-side, or follow-ahead, where follow-behind and side-by-side have been studied extensively [3]–[5]. In contrast, follow-ahead is more challenging because the robot must predict human intent and navigate proactively.

Prior follow-ahead work includes Moustris et al. [6], who used intention recognition for front-following. Nikdel et al. [7] combined DRL-based trajectory estimation with classical planners, referred to as LBGP. Mahdavian et al. [8] trained a non-autoregressive transformer for intention prediction. Obstacle and occlusion avoidance in follow-behind or side-by-side scenarios has been explored using lidar and cameras with PD and MPC controllers [9]–[14]; however, no existing follow-ahead method handles this problem.

We replicate the MCTS-DRL approach of [1], [2], the first follow-ahead method to address obstacle and occlusion avoidance. In this replication, we: (1) re-implement the framework from scratch in ROS2, including the MCTS planner, RL value function, and LSTM predictor; (2) deploy and evaluate the system on a QBot 2e robot using VICON for pose estimation and Cartographer for mapping; and (3) evaluate on additional trajectories to further probe follow-ahead behavior.

II. Methods

Unless otherwise specified, numerical constants such as turn angle, velocity, and training parameters are treated as hyperparameters tuned empirically.

Fig. 1. Real-world deployment of the MCTS-DRL follow-ahead system on a QBot robot tracking a person.

A. State, Observation, and Action Space

The system state is state $= (x_r, y_r, \theta_r, x_h, y_h, \theta_h)$, where x, y, θ denote 2D pose and subscripts r, h refer to robot and human. Each MCTS node represents one such state. The DRL model uses the relative observation $o = (x_h - x_r,\ y_h - y_r,\ \theta_h,\ \theta_r)$.

Six discrete actions are defined per time step: go straight, turn left θ, or turn right θ, each at two velocity levels. Using Dubins car dynamics, the robot's next pose under action $\psi \in \{-\theta, 0, \theta\}$ is

$$\begin{bmatrix} x_r' \\ y_r' \\ \theta_r' \end{bmatrix} = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix} + \begin{bmatrix} d\cos(\theta_r + \psi) \\ d\sin(\theta_r + \psi) \\ \psi \end{bmatrix} \tag{1}$$

assuming constant velocity V, giving d meters per step.

B. Reward Function

The reward is $r = \max(r_d + r_\alpha, -1)$, where $r_\alpha = (\alpha_{max} - \alpha)/\alpha_{max}$ penalizes angular deviation and r_d penalizes distance error

$$r_d = \begin{cases} -(1 - d_h) & d_{min} < d_h < d_{lo} \\ 0 & d_{lo} \le d_h \le d_{hi} \\ -c_d(d_h - 1) & d_{hi} < d_h < d_{max} \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

Here α is the angle between the person-robot vector and the person's heading, d_h is the human-robot distance, and $[d_{lo}, d_{hi}]$ defines the target following range.

C. DRL Training

An A2C agent is trained offline in an obstacle-free simulation [1] with randomized human trajectories and initial robot poses. The critic learns a value function $V(o)$ over the relative observation o, which is used to score nodes during MCTS tree expansion. Training optimizes the expected return $R = \sum_{i=0}^{n} \gamma^i r_i$ with discount factor γ.

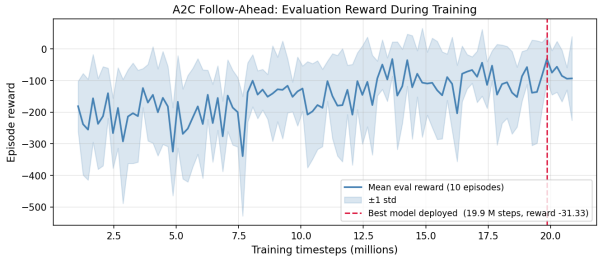

Fig. 2. Training curve of the RL value function used for MCTS node evaluation.

D. Human Action Prediction

An LSTM is trained to output a probability distribution over the human's next action given a short history of past poses. At each time step, the model takes the human's positions over the previous T_{lstm} seconds as input and produces probabilities over three actions: straight, left, and right. These probabilities are integrated into the UCB equation as P, biasing tree expansion toward branches where the human is more likely to move. Robot nodes are assigned a uniform prior $P = 1/6$. This allows the planner to follow confidently when the human moves predictably while remaining prepared for direction changes. In principle, any model that induces a distribution over future actions conditioned on past states can serve this role, with the LSTM being one practical choice.

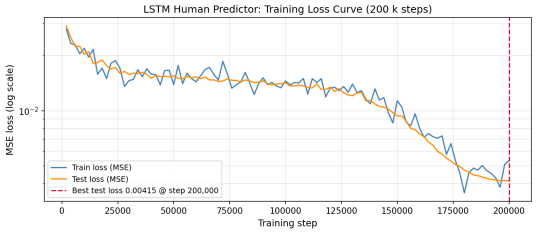

Fig. 3. LSTM training and test loss over 200k steps, achieving a best test MSE of 0.00415.

At each planning step, MCTS expands a search tree over a horizon T in a receding-horizon fashion. The root node encodes the current robot and human pose. Child nodes are generated and pruned if they collide with obstacles (via the occupancy map from Cartographer) or cause occlusion. Non-pruned nodes are assigned value $V(s)$ from the DRL model; pruned nodes receive -1. Human nodes are additionally weighted by action probabilities P from the LSTM predictor [2]. Node selection follows the Upper Confidence Bound

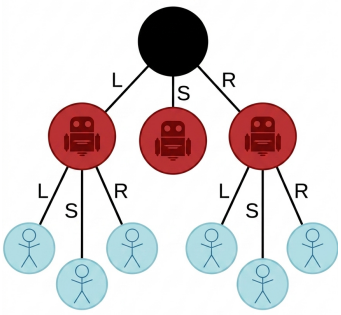

Fig. 4. MCTS tree expansion where blue and red nodes represent potential future positions of the robot and human respectively, with edges labeled by action: Left (L), Right (R), and Straight (S) (there are actions in practice).

Algorithm 1: The Proposed Approach: the tree expands over a receding horizon, selecting the most visited child as the optimal action.

Data: Robot current pose $= (x_r, y_r, \theta_r)^{t_0}$
Data: Human current pose $= (x_h, y_h, \theta_h)^{t_0}$
Data: RL model
Data: Occupancy map
Result: Best robot action
Root node $= (x_r, y_r, \theta_r, x_h, y_h, \theta_h)^{t_0}$;
for *0.15 second* **do**
  while *not leaf node* **do**
    for *child in children* **do**
      UCB $\leftarrow V/n + c\,P\sqrt{\ln n^p / n}$;
    Node $\leftarrow$ children(max UCB);
  New leaf node $\leftarrow$ expansion(Node);
  if *leaf node is safe* **then**
    $V \leftarrow$ evaluation(leaf node);
  else
    Delete the node;
  if *leaf node is human node* **then**
    $P \leftarrow$ Probability(leaf node);
  else
    $P \leftarrow 1/6$;
  Backpropagation(V);
Best action $\leftarrow$ most visited child;

E. MCTS-DRL Integration

$$\text{UCB} = P\left(\frac{w}{n^c} + c\sqrt{\frac{\log n^p}{n^c}}\right) \tag{3}$$

where $w = Q(o, a_i)$ is the node value, n^c and n^p are visit counts of the child and parent nodes, P is the LSTM-predicted action probability, and c balances exploration and exploitation. After tree expansion, the leaf node with the highest UCB evaluated at $c = 0$ is selected as the navigational goal and published as a velocity command.

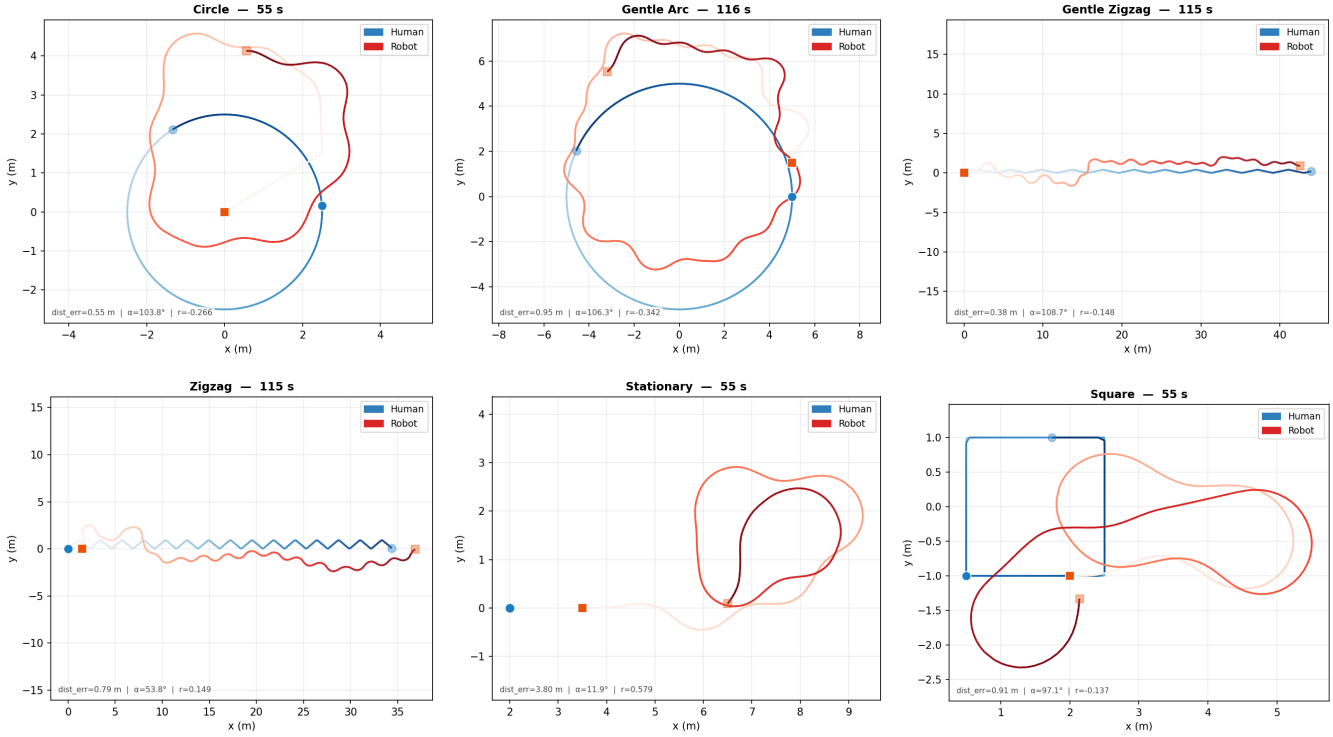

Fig. 5. Simulation trajectory plots across six test cases using the QBot. Top row, left to right: `circle`, `gentle_arc`, `gentle_zigzag`. Bottom row: `zigzag`, `stationary`, `square`. The `stationary` case exposes a known limitation of the current action space, addressed in future work, where the absence of a stay-still action causes the robot to overshoot and loop rather than hold position. The `square` case is inherently difficult as sharp 90° corners are an unusual and extreme trajectory that rarely occurs in natural human movement. Darker segments indicate more recent positions, with opacity decreasing toward earlier time steps.

F. Summary

Together, these three components form a closed loop at each time step. The RL model grounds each node with a learned sense of follow-ahead quality, while the LSTM biases expansion toward more probable human futures. The UCB criterion combines both into a single selection score, balancing exploration and exploitation. Obstacle and occlusion constraints are enforced separately by pruning unsafe nodes from the occupancy map, keeping collision avoidance independent of the learned components.

III. Experiments

We implement the MCTS-DRL framework from scratch in ROS2, replacing the original ROS1 codebase entirely. The system is deployed on a QBot robot, using VICON for real-time pose estimation and Cartographer for mapping. We evaluate on a set of trajectories designed to probe follow-ahead behavior under controlled conditions, extending beyond those tested in [1].

A. Implementation

The planning loop runs at 5 Hz. At each cycle, robot and human poses are ingested from VICON callbacks, a `FollowState` is constructed, an LSTM-based human action predictor outputs a distribution over {*straight, left, right*}, and `MCTSPlanner` runs for a 0.15 s wall-clock budget. The qBot has six actions: *straight*, *left*, *right*, and fast variants of each. Nodes are scored by a pretrained A2C value function; branches violating a displaced-circle safety boundary around the human are pruned. The best root action is selected by visit count and published as a `geometry_msgs/Twist` on `/cmd_vel`.

B. Simulation Setup

Before hardware deployment, all algorithm changes were validated in a fully closed-loop software simulation requiring only a CPU and ROS2 Humble. Scripted human motion was published by `fake_vicon.py`. This improves upon the original work's simulation, which evaluated policies using unconstrained instantaneous kinematics, our `fake_odom.py` node integrates `/cmd_vel` back into robot pose at 50 Hz while enforcing strict QBot hardware limits ($V_{\max}$=0.6 m/s, $\omega_{\max}$=0.5 rad/s, acceleration cap 0.5 m/s^2). This custom ramping ensures simulated dynamics faithfully match the physical platform, bridging a critical sim-to-real gap prior to deployment.

We added two more test cases. namely gentle zig-zag and gentle curve, to the original 5 test cases; results for `straight` and `gentle_arc` were not logged. Performance is measured by four metrics matching the evaluation criteria of [1]: mean distance error $e_d = \overline{|d_h - 1.5|}$ (m), mean angular offset $\bar{\alpha}$ (rad), mean reward $\bar{r} = \overline{r_d + r_\alpha}$, and in-cone rate η (% of cycles with $\alpha < 50°$).

C. Results

TABLE I
SIMULATION RESULTS PER TEST CASE (10 RUNS EACH, $d_{\text{TARGET}} = 1.5$ M, 75 S OPEN-ENVIRONMENT TRIALS, RANDOMISED START POSES FOR RUNS 2–10). e_d: MEAN DISTANCE ERROR (M); $\bar{\alpha}$: MEAN ANGULAR OFFSET; $\bar{r}$: MEAN REWARD PER STEP; η: % OF CYCLES WITH $\alpha < 50°$. VALUES ARE MEAN $\pm$ STD ACROSS RUNS.

Test case	e_d (m)	$\bar{\alpha}$ (°)	$\bar{r}$	η (%)
`stationary`	2.58 ± 0.40	29.2 ± 5.9	0.47 ± 0.09	84.7 ± 4.1
`zigzag`	0.84 ± 0.27	55.8 ± 7.6	0.05 ± 0.11	51.0 ± 3.2
`gentle_zigzag`	0.50 ± 0.16	80.7 ± 29.7	-0.15 ± 0.31	29.7 ± 25.8
`oscillate`	1.19 ± 0.28	86.3 ± 1.7	-0.20 ± 0.08	39.4 ± 6.0
`square`	0.74 ± 0.15	104.2 ± 6.4	-0.27 ± 0.11	19.8 ± 5.6
`circle`	1.04 ± 0.18	124.4 ± 7.1	-0.53 ± 0.06	6.6 ± 5.1
`gentle_arc`	1.03 ± 0.24	138.3 ± 12.4	-0.55 ± 0.11	2.9 ± 9.2

The system performs best when the human is slow-moving or stationary. `stationary` achieves the highest reward ($\bar{r} = 0.47$) and maintains α well below 50° in 84.7% of cycles ($\eta = 84.7\%$), confirming that the planner reliably acquires and holds a lead position when the human provides a stable target. The elevated e_d of 2.58 m reflects the robot overshooting the 1.5 m target distance rather than a directional failure; the robot is consistently ahead but at a larger separation than desired.

Performance degrades sharply on curved or oscillatory trajectories. `circle`, `gentle_arc`, and `square` all yield mean $\bar{\alpha}$ above 100°, corresponding to the robot falling behind the human rather than leading, and in-cone rates below 20%. Among the directional cases, trajectory smoothness correlates with orientation recovery: `gentle_zigzag` averages $\bar{\alpha} = 80.7°$ against `oscillate`'s 86.3° and `square`'s 104.2°, and achieves the lowest e_d of all directional cases (0.50 m). Although neither reaches the $\eta > 50\%$ threshold, the trend confirms that gradual heading changes give the planner more time to reposition before the human's trajectory diverges. The `zigzag` case is a partial exception: despite the direction reversals, the robot achieves $\eta = 51\%$ and a marginally positive reward (0.05), suggesting that the discrete nature of the zigzag pattern gives the planner just enough time to reposition between turns.

The consistent failure mode on continuous curves is that the receding-horizon planner commits to a trajectory branch before the human's heading change is detectable by the LSTM predictor, causing the robot to overshoot and then trail rather than lead.

D. Hardware

After simulation validation, experiments were conducted on a physical QBot 2e (Jetson Orin Nano 8 GB, ROS2 Humble) in an obstacle-free VICON lab at SFU. A VICON Vantage system tracked robot and human poses in real-time using a dedicated helmet marker set for the human and markers attached to the QBot. To integrate this external tracking into the ROS2 navigation stack, we developed a dedicated bridging node (`vicon_bridge.py`) that samples the VICON TF tree at 20 Hz and publishes `TransformStamped` topics for the planner. Furthermore, we configured a custom driver launch profile (`bringup_vicon.launch.py`) that pipes the `vrpn_mocap` data into a map-level Extended Kalman Filter (EKF). This setup provides robust, ground-truth map-to-odom corrections for the physical robot, successfully bypassing the need for noisy onboard SLAM during evaluation.

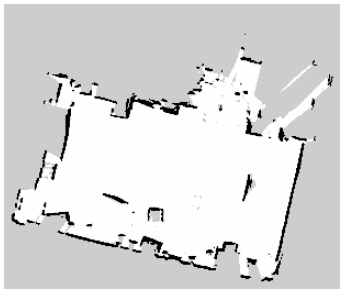

Fig. 6. Occupancy map of the VICON lab generated by Cartographer.

Pose streams were corrected by a 90° CCW rotation to align the VICON frame with the map frame, and Cartographer provided the occupancy map for localization as shown in Fig. 6. The robot replanned at 5 Hz and published Twist commands to `qbot_driver`. Due to lab time constraints, hardware trials were limited to open-loop walkthroughs without quantitative logging. Qualitatively, the robot successfully maintained a lead on straight and gentle-curve trajectories.

E. Replication Notes

We do not evaluate against standalone MCTS or DRL baselines. Given a sufficient rollout budget, MCTS with a hardcoded distance-to-target reward should produce comparable results since the planning structure dominates over the choice of node evaluator. The LSTM provides the clearest benefit when the human follows a consistent trajectory, concentrating expansion toward likely futures, but overall performance remains bottlenecked by the MCTS expansion budget regardless of the node evaluator or predictor used. Furthermore, our ROS2 implementation introduces necessary tree-search guards to prevent duplicate expansion, ensuring mathematically consistent node evaluation.

IV. FUTURE WORK

The most practical and immediate improvement is expanding the action space. The current formulation has no stay-still action and no rotate-in-place action, meaning the robot cannot wait when the person stops and cannot reorient without moving forward. Adding these two actions would cover the most common real-world failure cases with minimal changes to the existing framework.

In addition, both papers separate obstacle avoidance entirely from the learned components. The 2023 paper states explicitly that *"We first used DDQN to train, in an obstacle-free environment"* [1], and the 2025 paper follows the same structure, with the algorithm stating *"if leaf node is safe then V ← evaluation(leaf node)"* [2], implying the value function only ever scores geometrically safe nodes. This raises a natural question about the RL component specifically: if obstacle avoidance is handled entirely by geometric pruning, a

handcrafted reward would serve the same role with no training overhead and would be easier to tune and verify. The LSTM extension is more justifiable since it directly improves MCTS sample efficiency by concentrating expansion budget toward likely human futures.

V. Conclusion

We replicated the MCTS-DRL follow-ahead framework of [1], [2], re-implementing the full system from scratch in ROS2, including the MCTS planner, RL value function, and LSTM predictor. The system was deployed on a QBot 2e using VICON for pose estimation and Cartographer for mapping. The method integrates a pretrained RL value function and LSTM action predictor into MCTS tree expansion, handling obstacles through occupancy map pruning. Experiments on additional trajectories under controlled conditions support the viability of the approach for follow-ahead navigation, with performance degrading predictably on sharp turns and stationary cases as expected from the method's known limitations.

References

[1] S. Leisiazar, E. J. Park, A. Lim, and M. Chen, "An mcts-drl based obstacle and occlusion avoidance methodology in robotic follow-ahead applications," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.

[2] S. Leisiazar, S. R. Razavi Rohani, E. J. Park, A. Lim, and M. Chen, "Adapting to frequent human direction changes in autonomous frontal following robots," *IEEE Robotics and Automation Letters*, 2025.

[3] S. Sekiguchi, A. Yorozu, K. Kuno, M. Okada, Y. Watanabe, and M. Takahashi, "Uncertainty-aware non-linear model predictive control for human-following companion robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8316–8322.

[4] Q. Yan, J. Huang, Z. Yang, Y. Hasegawa, and T. Fukuda, "Human-following control of cane-type walking-aid robot within fixed relative posture," *IEEE/ASME Transactions on Mechatronics*, 2021.

[5] D. Jin, Z. Fang, and J. Zeng, "A robust autonomous following method for mobile robots in dynamic environments," *IEEE Access*, vol. 8, pp. 150 311–150 325, 2020.

[6] G. P. Moustris and C. S. Tzafestas, "Intention-based front-following control for an intelligent robotic rollator in indoor environments," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–7.

[7] P. Nikdel, R. Vaughan, and M. Chen, "Lbgp: Learning based goal planning for autonomous following in front," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3140–3146.

[8] M. Mahdavian, P. Nikdel, M. TaherAhmadi, and M. Chen, "Stpotr: Simultaneous human trajectory and pose prediction using a non-autoregressive transformer for robot following ahead," *arXiv preprint arXiv:2209.07600*, 2022.

[9] H. Yao, H. Dai, E. Zhao, P. Liu, and R. Zhao, "Laser-based side-by-side following for human-following robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2651–2656.

[10] J. Zou, "Predictive visual control network for occlusion solution in human-following robot," *Assembly Automation*, 2021.

[11] M. F. Aslan, A. Durdu, K. Sabanci, and M. A. Mutluer, "Cnn and hog based comparison study for complete occlusion handling in human tracking," *Measurement*, vol. 158, p. 107704, 2020.

[12] L. Pang, Y. Zhang, S. Coleman, and H. Cao, "Efficient hybrid-supervised deep reinforcement learning for person following robot," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 2, pp. 299–312, 2020.

[13] A. K. Ashe and K. M. Krishna, "Maneuvering intersections & occlusions using mpc-based prioritized tracking for differential drive person following robot," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1352–1357.

[14] H. Shin and S.-E. Yoon, "Optimization-based path planning for person following using following field," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 11 352–11 359.