LegNav: A Simulation Environment for Learning Social Navigation among Human Legs

Alberto Vaglio[1], Andrea Garulli[1], Antonio Giannitrapani[1], Renato Quartullo[2], and Tommaso Van Der Meer[1]

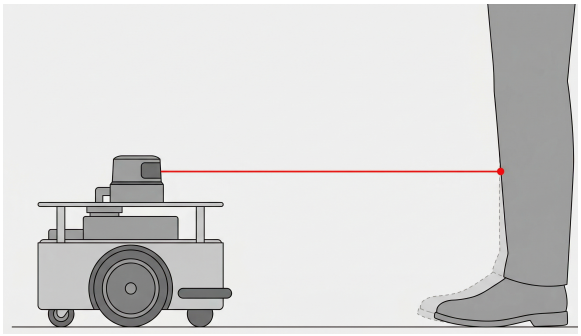
***Abstract*— Social robot navigation techniques based on reinforcement learning rely on simulations to train control policies. Standard simulators approximate pedestrians as simple circular shapes. Mobile robots using two-dimensional range sensors mounted near the ground detect individual moving legs and may miss protruding shoe shapes, instead of perceiving full bodies. This mismatch creates a gap between simulation and reality. To address this problem, this paper introduces LegNav, a simulation environment leveraging a non-slip gait model to describe realistic human leg motion. The simulator is used to train an end-to-end neural network architecture that processes the sensor readings taken from such environment to generate velocity commands. Simulation results show that policies trained with explicit leg dynamics achieve higher success rates and lower collision rates compared to some traditional methods. The trained policy is deployed on a TurtleBot 4 to carry out experiments in human-populated environments.**

I. Introduction and Related Work

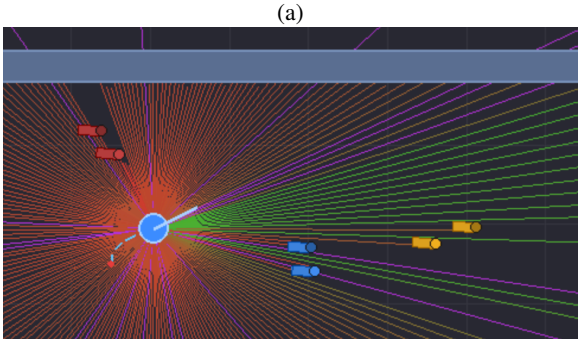
Social navigation represents a primary challenge for mobile robots operating in populated environments. Unlike static obstacle avoidance, social navigation requires the robot to handle dynamic human motion and respect interpersonal space. The robot must react to unpredictable pedestrian behaviors and adhere to implicit social norms.

Classical approaches to social navigation either rely on mathematical models imitating human motion or directly use the kinemtatic model of the robot in their formulation. The Social Force Model (SFM) [1] and the Headed Social Force Model (HSFM) [2] describe pedestrian motion through a combination of attractive goal forces, repulsive inter-agent forces, and repulsive obstacle forces. The difference between the two being that the latter also models the angular velocity and the orientation of each pedestrian. Velocity Obstacle (VO) methods and their reciprocal variants, such as ORCA [3], compute collision-free velocities by analyzing the relative velocity space of all agents. These models provide a strong foundation for simulating crowd behavior and for multi-robot navigation. However, using them as navigation controllers requires explicit tracking pipelines to estimate the precise positions and velocities of surrounding pedestrians. The Dynamic Window Approach (DWA) [4] samples feasible velocity commands within the kinematic constraints of the robot and selects the command that optimizes goal-seeking and obstacle avoidance. Model Predictive Path Integral (MPPI) [5] is a sampling-based variant of model predictive control capable of handling non-linear dynamics. This planner randomly samples multiple trajectories along a prediction horizon and scores them using a cost function. Finally, the selected trajectory (which is applied in a receding horizon fashion) is a weighted average of the sampled trajectories using the costs as weights. Despite their versatility, DWA and MPPI treat all range returns as generic obstacles and lack intrinsic mechanisms to understand social compliance. Moreover, all classical methods tend to get stuck in local minima.

To overcome these limitations, research shifted towards Deep Reinforcement Learning (DRL) techniques. Human-aware DRL methods tackle social navigation by incorporating the kinematic states of surrounding humans into the state representation of the robot. Early approaches utilize discrete action spaces and assume full observability of the



(a)



(b)

Fig. 1: (a) Side view of a mobile robot equipped with an ankle-height LiDAR. The beam intercepts the nearest boundary of the pedestrian's leg, while the shoe, protruding below the scan plane, remains undetected. (b) The simulation environment, featuring explicit human leg dynamics and realistic LiDAR returns from individual leg pairs.

[1]Department of Information Engineering and Mathematics, University of Siena, Siena, Italy. Email: {andrea.garulli, antonio.giannitrapani}@unisi.it and {alberto.vaglio, tommaso.vandermeer}@student.unisi.it

[2]Uninettuno University, Rome, Italy. Email: renato.quartullo@uninettunouniversity.net

crowd dynamics (positions and velocities). For example, Collision Avoidance DRL (CADRL) [6] frames pairwise collision avoidance as a sequential decision problem and uses a value network to estimate the value of future states. Socially-aware CADRL (SA-CADRL) [7] extends this concept to multi-agent settings. Social Attentive Reinforcement Learning (SARL) [8] incorporates attention mechanisms to assign varying importance weights to individual pedestrians. Subsequent works expand on these foundations to capture complex interactions. Relational graph learning approaches [9] model higher-order agent interactions and perform multi-step planning by predicting human trajectories. The Spatial-Temporal State Transformer [10] encodes global spatial interactions and temporal correlations among consecutive frames. The work in [11] abstracts the dynamic environment into a robot-centric velocity space model. A major limitation of all these human-aware approaches is their reliance on full state information (positions and velocities of all pedestrians). Providing explicit pedestrian trajectories to the DRL agent requires sophisticated perception and tracking systems. These systems are computationally expensive in real-world deployments.

End-to-end (E2E) DRL methods offer an alternative paradigm by mapping raw sensor data to robot actions. This approach eliminates the intermediate detection pipeline and reduces system complexity. Numerous works address 2D LiDAR navigation by processing raw inputs. Policies exist to navigate tight indoor spaces [12] and maintain personal safety around pedestrians [13]. NavRep [14] trains unsupervised representation models to encode sensor observations into compact states. To handle dynamic environments, recent frameworks process sequential observations to infer motion trends. Recurrent neural networks accumulate historical motion features [15], while spatio-temporal attention modules extract scene dynamics from consecutive scans [16].

DRL agents are usually trained in simulators to gather data faster. In standard practice, these simulators model humans as simple circular discs. The simulation engine performs both collision detection and LiDAR ray tracing on the boundaries of these discs. The agent learns to navigate around large unified cylinders that move with a continuous trajectory. This modeling choice creates a discrepancy with physical reality. Many mobile robots feature 2D LiDAR sensors mounted between 10 and 20 cm above the ground [17], [18]. As shown in Fig. 1a, at this scanning height, the horizontal laser plane intersects only the lower legs of nearby pedestrians. Most beams hit one of the two shins or pass through the gap between them. The robot never perceives the torso or the full body outline of the pedestrian. The sensor observes two small point clusters for each person [17]. These clusters move independently and alternate as the pedestrian walks. One leg remains stationary on the ground during the stance phase, while the other leg swings forward. This geometric pattern is absent in disc-based simulators. Furthermore, partial observability introduces a critical blind spot at close range. The LiDAR intercepts the leg cylinder but misses the foot geometry. A standard adult shoe extends forward from the ankle and protrudes below the scan plane. The front of the shoe remains undetected by the sensor. This invisible protrusion creates a significant collision risk during close-range social interactions.

The combination of alternating leg dynamics and invisible shoe protrusions poses a significant challenge in deploying trained policies in the real world. A policy trained to maintain a specific clearance from a disc will violate the personal space of a real pedestrian because the physical footprint is larger and irregularly shaped compared to the LiDAR return. Policies trained on circular discs expect continuous sensor returns and evaluate collisions based on the visible boundary. When deployed on a physical robot, these policies may fail to interpret the fragmented leg clusters and collide with the undetected feet. Modeling the motion of human legs in simulation is expected to improve training of end-to-end navigation policies based on 2D LiDAR data.

The contribution of this paper is the introduction of a mathematical model to simulate human leg motion. The associated simulation environment LegNav (shown in Fig. 1b) is implemented entirely in JAX and used to train an E2E neural network architecture denoted as CALF (Convolutional Attention for Leg Features). The network features are tailored to process sensor data in environments where human legs are explicitly modeled. The proposed approach is validated through simulation benchmarks and tested on real-world experiments, supported by video documentation.

II. Simulating leg dynamics

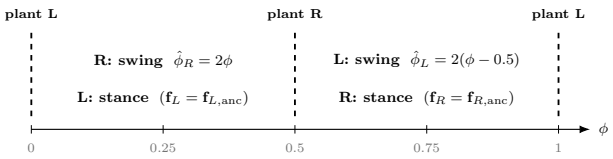
This section introduces the Non-Slip Gait (NSG) model to simulate humans' leg dynamics. The model represents pedestrians as two moving feet rather than a single disc. Each foot is assigned a rectangular footprint to define the shoe geometry. To mimic real sensor data, the legs are represented as small discs centered at the heel.

A common approach for animating legs expresses foot positions as sinusoidal offsets from the body center [19]. This formulation suffers from a kinematic inconsistency known as foot-skating. During the animation, stance feet translate rigidly with the body rather than remaining planted on the ground. To avoid this artifact, the NSG model introduces two distinct phases for leg motions: stance and swing. During the stance phase, the foot is anchored to a touchdown position and cannot move. During the swing phase, the foot is free to move.

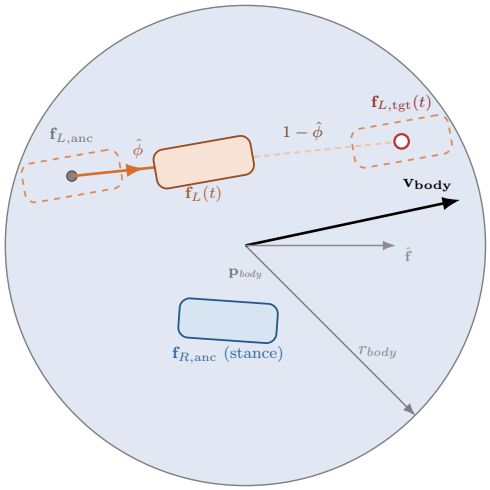
The pedestrian base moves following the HSFM [2], which models the body pose, and its linear and angular velocities. A continuous gait phase variable $\phi \in [0, 1)$ coordinates the switching between the two phases of motion of the leg. The gait phase advances at a cadence c, which is proportional to the body speed. The cadence is calculated as:

$$c(t) = f_{\max} \text{clip}\left(\frac{|\mathbf{v}_{\text{body}}(t)|}{v_{\text{ref}}}, 0.3, 1.0 \right) \tag{1}$$

where $f_{\max}$ is the maximum stepping frequency, v_{ref} is a reference walking speed and $\mathbf{v}_{\text{body}}(t)$ is the linear velocity



(a)



(b)

Fig. 2: (a) The gait phase $\phi \in [0, 1)$ advances continuously. At each half-cycle crossing, the corresponding foot plants and its anchor is updated. (b) During swing, the foot interpolates from its anchor toward a dynamic target.

of the pedestrian's body at time t, regulated by the HSFM. The phase update is expressed as:

$$\phi(t+1) = (\phi(t) + c(t)\Delta t) \bmod 1.0 \tag{2}$$

One complete gait cycle consists of two symmetric half-periods. If $\phi \in [0, 0.5)$, the left foot is planted and the right foot swings. If $\phi \in [0.5, 1.0)$, the right foot is planted and the left foot swings. The gait phase is mapped to the progress of each foot swing $\hat{\phi} \in [0, 1)$, running from 0 at lift-off to 1 at touchdown. Fig. 2a shows the temporal evolution of the gait phase. The swinging foot position $\mathbf{f}(t)$ interpolates between the planted anchor $\mathbf{f}_{\text{anc}}$ and a dynamically updated target $\mathbf{f}_{\text{tgt}}(t)$ (as shown in Fig. 2b) as:

$$\mathbf{f}(t) = \mathbf{f}_{\text{anc}}(1 - \hat{\phi}(t)) + \mathbf{f}_{\text{tgt}}(t)\hat{\phi}(t) \tag{3}$$

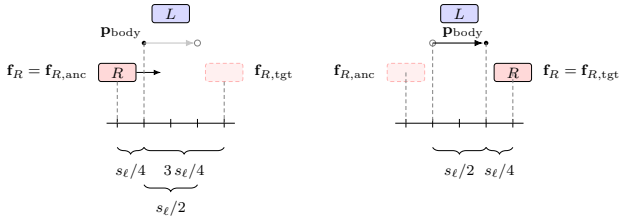
The foot target is updated based on the pedestrian body position $\mathbf{p}_{\text{body}}(t)$ and its body heading $\alpha_{\text{body}}(t)$. The target is defined as:

$$\mathbf{f}_{\text{tgt}}(t) = \mathbf{p}_{\text{body}}(t) + \hat{\mathbf{f}}(t)\ell(\hat{\phi}(t)) \pm \hat{\mathbf{l}}(t)\frac{w_{\text{hip}}}{2} \tag{4}$$

where $\hat{\mathbf{f}}(t) = [\cos(\alpha_{\text{body}}(t)), \sin(\alpha_{\text{body}}(t))]^T$ and $\hat{\mathbf{l}}(t) = [-\sin(\alpha_{\text{body}}(t)), \cos(\alpha_{\text{body}}(t))]^T$ are the frontal and left-lateral unit vectors with respect to the pedestrian heading, and w_{hip} is the hip width. The sign of the lateral offset $\hat{\mathbf{l}}(t)\frac{w_{\text{hip}}}{2}$ depends wether the target is computed for the left foot (positive) or right foot (negative). The function $\ell(\hat{\phi}(t))$ determines the forward reach based on the stride length s_l:

$$\ell(\hat{\phi}) = \left(\frac{3}{4} - \frac{1}{2}\hat{\phi}(t)\right) s_l(t), \quad s_l(t) = \frac{|\mathbf{v}_{\text{body}}(t)|}{c(t)}. \tag{5}$$

The swing process is illustrated in Fig. 3. During straight-line motion at a constant speed, the body and each foot advance by a stride s_ℓ over one full cycle. Within a single step, the body moves by $s_\ell/2$. During the same step, the swinging foot covers a distance s_ℓ and overtakes the body. The target is measured from the *current* body center, causing its relative position to change during the swing phase. At lift-off ($\hat{\phi} = 0$), the swing foot starts $s_\ell/4$ *behind* the body center. After an advancement of one stride, the foot reaches a position $-s_\ell/4 + s_\ell = 3s_\ell/4$ in front of the initial center location. At touchdown ($\hat{\phi} = 1$), the body completes an advancement of $s_\ell/2$. As a result, the landing point lies $(3/4 - 1/2)s_\ell = s_\ell/4$ in front of the updated body center.



(a) $\hat{\phi} = 0$ (b) $\hat{\phi} = 1$

Fig. 3: Swing-foot target offset at (a) the start and (b) the end of a step. Distances are measured along the walking direction from the current body center.

Updating the target at every simulation step allows the swing trajectory to follow body turns and speed variations. The target position changes relative to the body center during the swing phase, as detailed in Fig. 3.

The NSG model is embedded into LegNav [1], a custom 2D simulation environment written in Python using the JAX library [20]. The simulator performs 2D LiDAR ray tracing on the individual leg discs but detects collisions using the foot shapes. The robot implemented in the simulator is a differential drive robot.

III. The Navigation Policy Architecture

Differentiating between moving agents and static obstacles directly from raw LiDAR scans is not a simple task. To capture the alternating leg dynamics directly from raw LiDAR scans, the Convolutional Attention for Leg Features (CALF) architecture is developed. The network maps raw sensor measurements to robot velocity commands in an E2E fashion. The architecture combines a one-dimensional Convolutional Neural Network (1D-CNN), self-attention mechanisms, and a Multi-Layer Perceptron (MLP). The full structure is visible in Fig. 4.

[1]Available at `https://github.com/otr-ebla/LegNav-Sim`

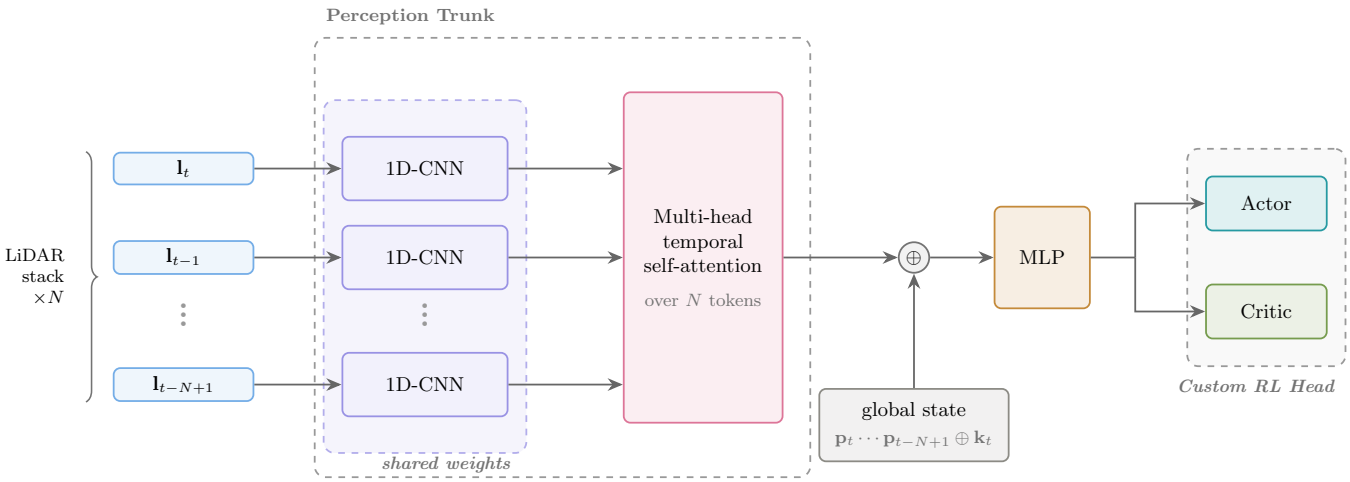


Fig. 4: CALF end-to-end policy architecture processing stacked 2D LiDAR frames via a shared CNN and temporal self-attention. The latest N LiDAR scans $\{\mathbf{l}_k\}_{k=t,\dots,t-N+1}$ are fed into the perception trunk, producing the extacted LiDAR features. These are concatenated with the last N goal positions in the robot frame $\{\mathbf{p}_k\}_{k=t,\dots,t-N+1}$ along with the kinematic vector $\mathbf{k}_t$ containing the current linear and angular velocities of the robot along with its maximum velocity. This concatenated vector is processed into an MLP and then processed into the individual actor and critic heads to produce the action and the state value.

The 1D-CNN encoder processes a stack of consecutive LiDAR scans. The encoder uses shared weights across the temporal stack to extract geometric features. A multi-head temporal self-attention module processes the extracted features across the frames. The attention mechanism infers leg motion and dynamic states without explicit pedestrian tracking modules. The global state vector contains the goal coordinates and the robot kinematic state. This vector bypasses the perception trunk and is fused with the spatio-temporal LiDAR features produced by the attention module. The combined representation is finally processed by the MLP to generate the final action distribution.

IV. Reward Function

The total reward function combines navigation objectives with social compliance incentives. The formulation avoids complex equations and relies on a sum of individual components. The total per-step reward r_t at each time step t is calculated as the sum of all the individual weighted per-step terms:

$$r_t = r_{\text{prog},t} + r_{\text{step},t} + r_{\text{smooth},t} + r_{\text{speed},t} + r_{\text{heading},t} + r_{\text{comfort},t} + r_{\text{yield},t} \tag{6}$$

The per-step terms shape specific aspects of the desired navigation behavior:

- Progress reward (r_{prog}): rewards the reduction of the Euclidean distance to the goal.
- Step penalty (r_{step}): imposes a constant cost per time step to discourage idle behavior.
- Smoothness penalty (r_{smooth}): penalizes changes in angular velocity between consecutive steps to suppress oscillatory steering.
- Speed reward (r_{speed}): mitigates the step penalty in proportion to the forward velocity of the robot. This term incentivizes motion at higher speeds.
- Heading reward (r_{heading}): rewards forward motion directed toward the goal. It couples the speed incentive to goal alignment.
- Comfort penalty (r_{comfort}): penalizes intrusions into pedestrian personal space. The penalty scales with the robot speed to assign higher costs to fast intrusions.
- Yield reward (r_{yield}): encourages the robot to yield when a pedestrian crosses its path. A frontal yield zone is defined based on a specific radius and field of view. If a pedestrian occupies this zone, the term rewards robot velocities below a stopping threshold and penalizes higher speeds based on the pedestrian proximity. The standard speed reward is suppressed during this event.

Terminal episode events define additional reward terms, which are added to the total reward at the end of the episode. Reaching the goal yields the highest positive bonus. Terminal failures receive penalties based on the event type:

- Active collision penalty: penalizes contacts with pedestrians when the robot moves above the stopping threshold and the pedestrian occupies the frontal yield zone. This event indicates a failure to yield.
- Obstacle collision penalty: penalizes collisions with static obstacles or walls.
- Timeout penalty: penalizes the failure to reach the goal within the maximum episode duration.
- Passive collision penalty: penalizes contacts occurring when the robot is stationary or when the pedestrian approaches from outside the frontal yield zone. The penalty is smaller because the robot lacks full respon-

sibility for the contact.

V. Experimental evaluation

The CALF policy is trained using Proximal Policy Optimization (PPO) [21] in different scenarios. To isolate the effect of the leg model, a baseline policy is trained under the exact same conditions using the standard disc approximation of pedestrians. The test environment incorporates NSG-based leg dynamics, accounts for shoe geometry in collision detection, and allows humans to react to the robot, providing more realistic conditions for evaluating the trained policies. The NSG model is implemented using the following parameters: $f_{\max} = 3.5\,\mathrm{s}^{-1}$, $v_{\mathrm{ref}} = 2.5\,\mathrm{m/s}$, $w_{\mathrm{hip}} = 0.3\,\mathrm{m}$. The legs of pedestrian have a radius of $0.08\,\mathrm{m}$, and the shoes rectangles have a length of $0.3\,\mathrm{m}$ and a width of $0.1\,\mathrm{m}$.

The policies are evaluated in unseen test scenarios (implementing leg dynamics) for 1000 episodes using specific metrics. Success Rate (SR) defines the fraction of episodes where the robot reaches the goal without colliding within a predefined time limit. Active Collision Rate (ACR) defines the fraction of episodes ending in collisions possibly caused by the robot (i.e., the robot velocity is higher than a certain threshold during collision). Passive Collision Rate (PCR) captures the fraction of episodes ending in collisions with humans that are mostly due to humans' behavior (i.e., the robot velocity is lower than a certain threshold upon collision). Space Compliance (SC) is the fraction of time steps in which the robot keeps a distance greater than 0.5 m from humans. Table I summarizes the performance comparison. The residual episode outcomes are all timeouts, i.e., the episode ended because no termination occurred within the time limit. The preference for each metric is expressed with an up or down arrow.

TABLE I: Comparison of policies trained with and without legs

Method	SR (%) ↑	ACR (%) ↓	PCR (%) ↓	SC (%) ↑
CALF (discs)	73.1	24.5	2.3	76.0
CALF (legs)	95.1	3.3	0.9	83.5

The policy trained with the NSG model achieves a success rate of 95.1%, an active collision rate of 3.3% and a passive collision rate of 0.9%. The policy trained with the disc model achieves a success rate of 73.1%, an active collision rate of 24.5%, and a passive collision rate of 2.3%. Regarding more social metrics, the policy trained in the environment modeling the legs dynamics also achieves a higher space compliance.

The proposed policy is compared against classical model-based planners, DWA [4] and MPPI [5], and learning-based baselines, such as TAGD [16] and a simple E2E policy with a three layer MLP, denoted as VanillaE2E. The learning-based baselines have been trained in LegNav using the NSG model for legs dynamics, shoes rectrangles for collisions, and using the same reward function employed for CALF. Table II reports the results of this comparison. CALF achieves the highest success rate (95.1%) and the lowest passive collision rate (0.9%). DWA registers the lowest active collision rate (0.9%) but reaches a lower success rate (77.7%) due to timeouts caused by conservative behavior. MPPI completes 70.1% of the episodes. Among the learning-based baselines, TAGD shows a high success rate (88.2%) and the highest space compliance (85.0%), but it records an active collision rate (7.3%) higher than CALF. VanillaE2E lacks the specialized architecture and registers the highest active collision rate (32.5%) and the lowest success rate (64.5%). These results indicate that CALF provides an optimal balance between navigation success and safety.

TABLE II: Comparison with navigation baselines

Method	SR (%) ↑	ACR (%) ↓	PCR (%) ↓	SC (%) ↑
DWA	77.7	0.9	1.2	77.3
MPPI	70.1	5.5	1.2	81.7
TAGD	88.2	7.3	1.5	85.0
Vanilla E2E	64.5	32.5	2.6	72.3
CALF (legs)	95.1	3.3	0.9	83.5



Fig. 5: Real-world deployment experiments on a TurtleBot 4 platform.

The trained NSG policy is deployed zero-shot on a physical TurtleBot 4 platform, equipped with an ankle-height LiDAR sensor. The robot demonstrates proactive social awareness and safe yielding behaviors in real indoor environments with human participants (Fig. 5). The compliant yielding behavior is triggered to guarantee safety in narrow corridors. A video of the real-world experiments is available at `https://www.youtube.com/watch?v=P6gFTvi3k7w`.

VI. Conclusion

A new simulation environment for social robot navigation has been introduced. The LegNav simulator replicates the alternating leg patterns and shoe protrusions observed by ankle-height LiDAR sensors in 2D environments. An E2E architecture mapping LiDAR sensor readings to velocity commands has been trained in LegNav. Numerical results have shown that training policies with explicit leg dynamics

improves the success rate and reduces collision rates compared to traditional disc-based training. Comparisons against classical planners and state-of-the-art learning baselines confirm that the proposed approach achieves a good balance between navigation success and safety. The policy trained in LegNav has been deployed on a TurtleBot 4 and tested in a real-world environment.

Several improvements can be foreseen for the proposed approach. More accurate modeling of pedestrian sharp turns would improve realism of human behaviors. The training process may be enhanced by adding penalty terms on linear acceleration to improve pedestrian comfort. Quantitative evaluation of real-world experiments are the subject of ongoing work.

REFERENCES

[1] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, 1995.

[2] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, and D. Prattichizzo, "Walking ahead: The headed social force model," *PLOS ONE*, vol. 12, no. 1, p. e0169734, 2017.

[3] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research* (C. Pradalier, R. Siegwart, and G. Hirzinger, eds.), (Berlin, Heidelberg), pp. 3–19, Springer Berlin Heidelberg, 2011.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[5] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[6] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 285–292, 2017.

[7] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350, 2017.

[8] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6015–6022, 2019.

[9] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10007–10013, 2020.

[10] Y. Yang, J. Jiang, J. Zhang, J. Huang, and M. Gao, "St^2: Spatial-temporal state transformer for crowd-aware autonomous navigation," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 912–919, 2023.

[11] D. Martinez-Baselga, L. Riazuelo, and L. Montano, "Rumor: Reinforcement learning for understanding a model of the real world for navigation in dynamic environments," *Robotics and Autonomous Systems*, vol. 191, p. 105020, 2025.

[12] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1140–1146, 2021.

[13] J. Jin, N. M. Nguyen, N. Sakib, D. Graves, H. Yao, and M. Jagersand, "Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2d laser scans," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6979–6985, 2020.

[14] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7829–7835, 2021.

[15] W. Zhu and M. Hayashibe, "Learn to navigate in dynamic environments with normalized lidar scans," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7568–7575, 2024.

[16] J. de Heuvel, X. Zeng, W. Shi, T. Sethuraman, and M. Bennewitz, "Spatiotemporal attention enhances lidar-based robot navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4202–4209, 2024.

[17] K. O. Arras, O. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2d range data," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3402–3407, 2007.

[18] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 726–733, 2015.

[19] R. Boulic, N. M. Thalmann, and D. Thalmann, "A global human walking model with real-time kinematic personification," *The visual computer*, vol. 6, no. 6, pp. 344–358, 1990.

[20] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. Version 0.3.13. Available at `http://github.com/jax-ml/jax`.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.