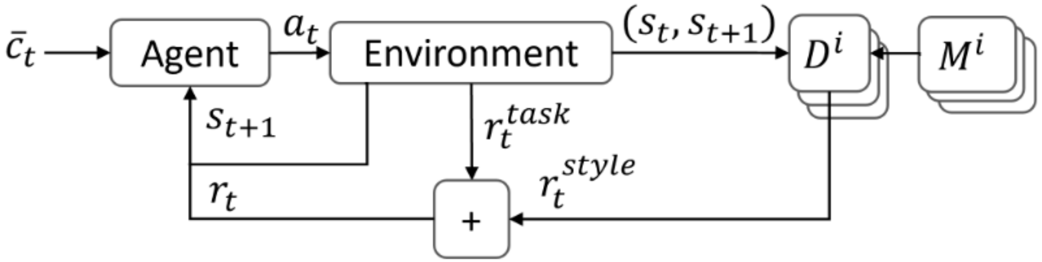


Fig. 2. Multi-AMP overview: The discriminator predicts a style reward s_t^{style} which is high if the policy's behavior is similar to the motions of the motion database M^i, by distinguishing between state transitions (s_t, s_{t+1}) of both sources. The style reward is added to the task reward, which finally leads to the policy fulfilling the task while applying the motion data's style.

Require: $M = \{M_i\}, |M| = n$ (n motion data-sets)

$\pi \leftarrow$ initialize policy
$V \leftarrow$ initialize Value function
$[\mathcal{B}] \leftarrow$ initialize n style replay buffers
$[D] \leftarrow$ initialize n discriminators
$\mathcal{R} \leftarrow$ initialize main replay buffers
while not done **do**
  for trajectory i = 1, ..., m **do**
    $\tau^i \leftarrow \{(c_t, c_s, s_t, a_t, r_t^G)_{t=0}^{T-1}, s_T, g\}$ roll-out with π
    $d \leftarrow$ style-index of τ^i (encoded in c_s)
    if M^d is not empty **then**
      for t = 0, ..., T-1 **do**
        $d_t \leftarrow D^d(\phi(s_t), \phi(s_{t+1}))$
        $r_t^{style} \leftarrow$ according to Eq. 2
        record r_t^{style} in τ^i
      end for
      store d_t in $\mathcal{B}^d$ and τ_i in $\mathcal{R}$
    end if
  end for
  for update step = 1, ..., $n_{updates}$ **do**
    for d = 0, ..., n **do**
      $b^{\mathcal{M}} \leftarrow$ sample batch of K transitions $\{s_j, s'_j\}_{j=1}^K$ from $\mathcal{M}^d$
      $b^{\pi} \leftarrow$ sample batch of K transitions $\{s_j, s'_j\}_{j=1}^K$ from $\mathcal{B}^d$
      update D^d according to Eq. 1
    end for
  end for
  update V and π (standard PPO step using $\mathcal{R}$)
end while

Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning

TABLE I

TASK-REWARDS.

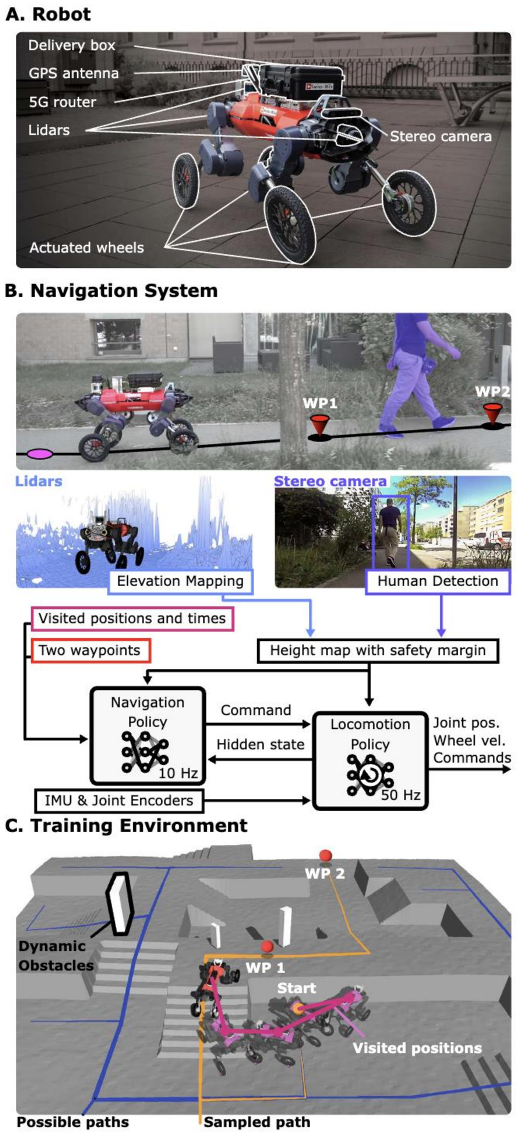
All tasks	formula	weight
r_τ	$\|\|\tau\|\|^2$	-0.0001
$r_{\dot{q}}$	$\|\|\dot{q}\|\|^2$	-0.0001
$r_{\ddot{q}}$	$\|\|\ddot{q}\|\|^2$	-0.0001
4-legged locomotion		
$r_{lin\ vel}$	$e^{\|\|\dot{x}_{target,\ xy} - \dot{x}\|\|^2/0.25}$	1.5
$r_{ang\ vel}$	$e^{\|\|\omega_{target,\ z} - \omega\|\|^2/0.25}$	1.5
Ducking		
r_{duck}	$e^{0.8 * \|x_{goal} - x\|}$	2
Stand-up	see Tab. II	

TABLE II

REWARDS FOR AOW STANDING UP, SITTING DOWN, AND NAVIGATING WHILE STANDING

symbols	description	
$q^{robot} \in \mathbb{H}$	Robot base-frame rotation	
$p^{robot} \in \mathbb{R}^3$	Robot base-frame position	
q	Joint DOF positions (excl. wheels)	
q_{hl}	Hind-Leg DOF position	
α	$\angle$(robot-x axis, world z axis)	
f	Feet on ground (binary)	
s	Standing robots (binary)	
stand-up	formula	weight
r_α	$\frac{\pi/2-\alpha}{\pi/2}$	2
r_{height}	p_z^{robot}	3
r_{feet}	f	-2
r_{wheels}	$\boxed{\sum \dot{q}^2_{front\ wheels} * (1-f)}$	$\boxed{-0.003}$
$r_{shoulder}$	$\|\|q_{shoulder}\|\|^2$	-1
$r_{stand\ pose}$	$exp(-0.1 * \|\|q_{hl} - q_{0,\ hl}\|\|^2)$	1
sit-down		weight
$r_{un-stand}$	$max(\frac{\pi/2-\alpha}{\pi/2} * 3, 0)$	-3
$r_{sit-down}$	$\frac{min(\alpha, \pi/2)}{\pi/2}$	2.65
$r_{dof\ vel}$	$\|\|\dot{q}\|\|^2$	-0.015
$r_{dof\ pos}$	$\exp(-0.5 * \|\|q_0 - q\|\|^2) * \frac{\alpha}{\pi/2}$	3
navigation		weight
$r_{track\ lin}$	$exp(-4 * \|\|\dot{x}_{des} + \dot{p}^{robot}_{local,z}\|\|^2) * s$	2
$r_{track\ ang}$	$exp(-4 * \|\|\omega_{des} - \omega^{robot}_{local,x}\|\|^2) * s$	2

Learning Robust Autonomous Navigation and Locomotion for Wheeled-Legged Robots



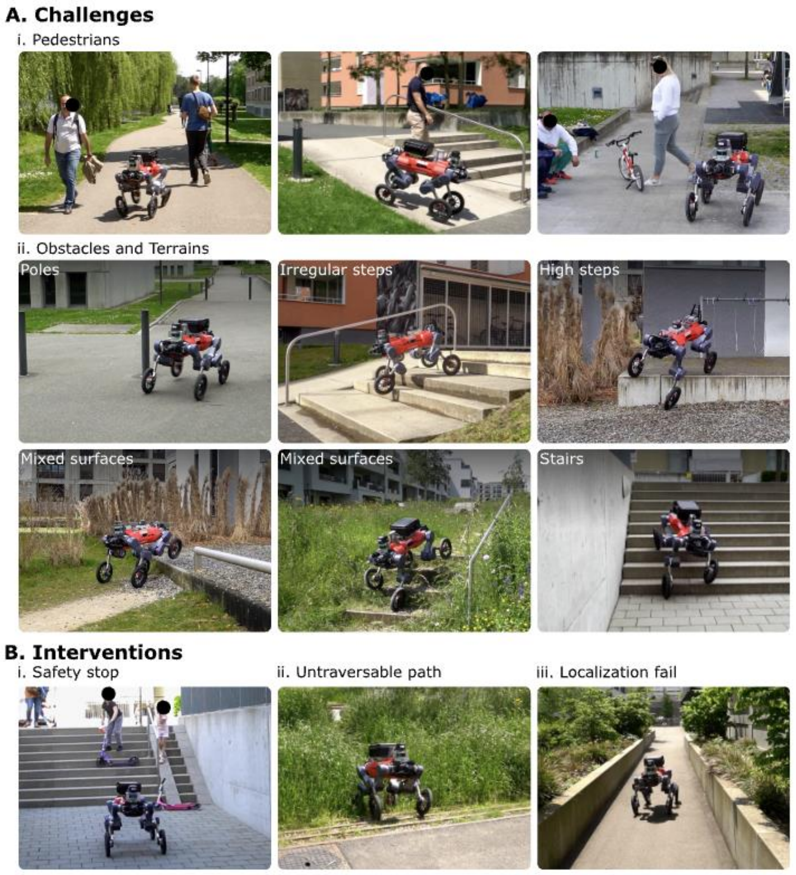


Fig. 4. Challenges in the populated urban environment. **(A)** The urban environment presents various obstacles. Some have to be avoided, such as pedestrians or poles, and others can be traversed, such as stairs or steps. **(B)** We had to intervene and stop the mission in these three cases.

Low-level Policy Rewards

r_l is modified from the reward terms by Miki et al. [16]. r_l is defined with the linear combination of the following reward terms.

The linear velocity tracking reward encourages the policy to follow a desired horizontal velocity (velocity in xy plane) command:

$$r_{lv} := \begin{cases} 2.0\exp(-2.0 \cdot ||v_{xy}^{body}||^2), & \text{if } |v_{des}| < 0.05 \\ \exp(-2.0||v_{xy}^{body} - v_{des}||^2) + v_{des} \cdot v_{xy}^{body}, & \text{otherwise} \end{cases}, \tag{14}$$

where $v_{des} \in \mathbb{R}^2$ is the desired horizontal velocity.

We also defined a reward to encourage the policy to follow a desired yaw velocity command:

$$r_{av} := \exp(-2.0(\omega_z^{body} - \omega_{des})^2). \tag{15}$$

As we aim for stable base motions, we defined a penalty for the body velocity in directions not part of the command:

$$r_{bm} := -1.25(v_z^{body})^2 - 0.4|\omega_x^{body}| - 0.4|\omega_y^B|. \tag{16}$$

We also penalized the angle between the z-axis of the world and the z-axis of the robot's body to maintain level body pose:

$$r_{ori} = \arccos(R_b(3,3))^2, \tag{17}$$

where $R_b(3,3)$ is the last element of the rotation matrix representation of the body orientation. We also motivated the policy to keep the height of the robot's base above the ground (h_{base}) around 0.55 m with the tolerance of 0.05 m:

$$r_h = max(0.0, |h_{base} - 0.55| - 0.05). \tag{18}$$

Regularization Rewards

We used various regularization rewards. We penalized the joint torques to prevent damaging joint actuators during deployment and to reduce energy consumption ($\tau \propto$ electric current):

$$r_\tau := -\textstyle\sum_{i \in joints}||\tau_i||^2. \tag{19}$$

we also penalized joint velocity and acceleration to avoid vibrations:

$$r_s = -c_k \sum_{i=1}^{12}(\dot{q}_i{}^2 + 0.01\ddot{q}_i{}^2), \tag{20}$$

where $\dot{q}_i$ and $\ddot{q}_i$ are the joint velocity and acceleration, respectively.

The magnitude of the first and second order finite difference derivatives of the target joint positions are penalized such that the generated joint trajectories become smoother:

$$r_s = -c_k \sum_{i=1}^{12}((q_{i,t,des} - q_{i,t-1,des})^2 + (q_{i,t,des} - 2q_{i,t-1,des} + q_{i,t-2,des})^2), \tag{21}$$

where $q_{i,t,des}$ is the joint target position of joint i at time step t.

We enforced soft position constraints in the joint space. To avoid the knee joint flipping in the opposite direction, we give a penalty for exceeding a threshold:

$$r_{jc,i} = \begin{cases} -(q_i - q_{i,th})^2, & \text{if } q_i > q_{i,th} \\ 0.0 & \text{otherwise} \end{cases}, \tag{22}$$

$$r_{jc} = \sum_{i=1}^{12} r_{jc,i}, \tag{23}$$

where $q_{i,th}$ is a threshold value for the ith joint. We only set thresholds for the knee joint.

Contacts with the environment were penalized except for the wheels:

$$\boxed{r_{bc} := -|I_{c,body} \backslash I_{c,wheel}|. \qquad (24)}$$

Not terminating was densely rewarded:

$$r_{h,surv} := 1.0 \quad \text{while not terminated.} \tag{25}$$

For the low-level policy, we introduced an additional gait-tracking reward, defined as

$$r_{gait} := 0.1 \cdot \sum_{i \in 0,1,2,3} \mathbb{1}(fc(i) = fc(i)_g), \quad \textbf{(27)}$$

where $fc_(i)$ denotes the desired contact state of the i-th foot and $fc_(i)_g$ is the target contact state given by the high-level policy.

Learned Action Space for Gait-generating High-level Policy

For the gait commanding high-level policy, we had to implement a special action space. Exploring the space of gait parameters with the commonly used Gaussian distribution can be inefficient because not all the real-valued vectors can represent feasible gaits, and the feasible parameters can be sparsely distributed. To improve exploration and accelerate learning, we use a learned gait generator as the action space of the high-level policy.

Existing works have proposed using generative models such as Variational Autoencoder (VAE)s [75, 76] or a normalizing flow [74] to transform the action distribution into a different, possibly multi-modal, distribution. Wenxuan et al. [75] and Allshire et al. [76] proposed to pre-train generative models with existing motion data for higher sample efficiency.

Similarly, we construct a learned latent action space with a Real-NVP model [74] that generates gait patterns from a Beta distribution. We chose RealNVP instead of VAE [77] because the RealNVP can be updated during the RL update by policy gradient thanks to its invertibility [74, 78].

We construct a stochastic policy $\pi(a|s)$ by two neural network modules in series. Firstly, an MLP outputs parameters for the Beta distribution that serves as a base distribution. Then follows an invertible normalizing flow layer to get $a = f_\psi(z)$, where $z \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$. f_ψ denotes a RealNVP. We can directly use the RealNVP policy instead of Gaussian policies within RL algorithms since it is possible to compute the log-likelihood of the action by

$$\log(\pi(a|s)) = \log(p_z(f_\psi^{-1}(a))) + \log\left(\left|\det\left(\frac{\partial f_\psi^{-1}(a)}{\partial a^T}\right)\right|\right). \quad \textbf{(28)}$$

The RealNVP layers are pre-trained to generate gait parameters from a uniform distribution. It is trained by minimizing the log-likelihood:

$$\mathbb{E}_x\left\{-\log(p_z(f_\psi^{-1}(x))) - \log\left(\left|\det(\partial f_\psi^{-1}(x)/\partial x^T)\right|\right)\right\}, \quad \textbf{(29)}$$

where x is sampled uniformly from known gait parameters.

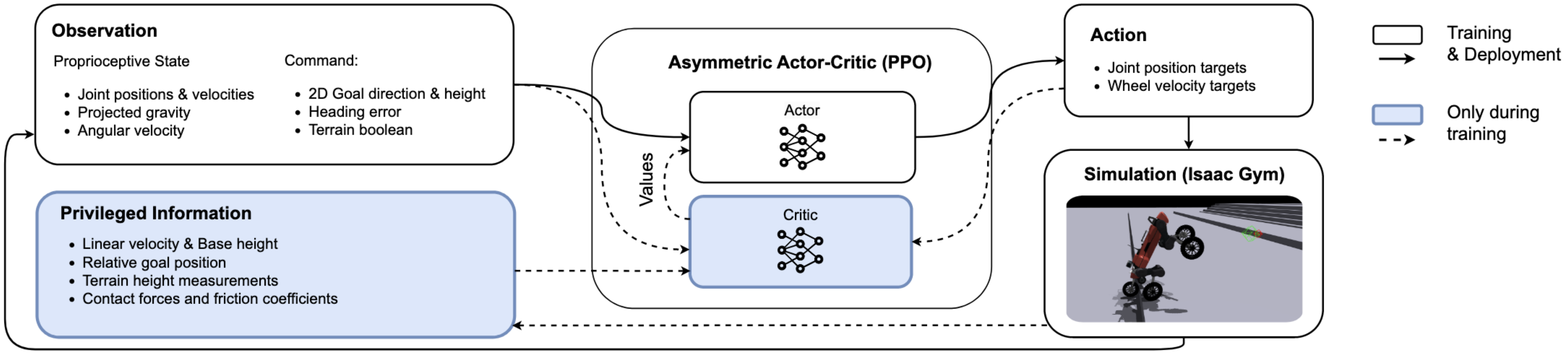


Fig. 2: **System Overview during Training and Deployment:** At every training step, the algorithm receives the observation and privileged information. The actor outputs an action for the next simulation step. During deployment, the actor receives only the observation and outputs an action for the robot to execute.

TABLE I: Observation & Privileged Information.

Symbol	Observation	Units	Coeff.	Size	Noise (%)
$\dot{\vec{\theta}}$	Angular velocity	rad/s	0.25	3	±20
$\vec{\gamma}$	Projected gravity	N/A	1.0	3	±5
$\vec{g_{dir}}$	Direction to goal	N/A	1.0	2	±0
θ_{err}	Heading error	rad	1.0	1	±0
h_{target}	Height command	m	1.0	1	±0
b	Terrain boolean	N/A	1.0	1	±0
$\vec{q}$	Joint positions	rad	1.0	4	±1
$\dot{\vec{q}}$	Joint velocities	rad/s	0.05	6	±150
$\vec{a_{last}}$	Last action	rad & rad/s	1.0	6	±0
Symbol	**Privileged Information**	**Units**	**Coeff.**	**Size**	**Noise (%)**
v_x	Linear velocity	m/s	0.25	1	-
h	Base height	m	1.0	3	-
$\vec{g_{rel}}$	Relative goal position	m	1.0	3	-
$\mathbf{H_{terrain}}$	Terrain height	m	5.0	187	-
$\vec{f_{wheels}}$	Contact forces	N	0.01	6	-
μ	Friction coefficient	N/A	1.0	1	-

TABLE II: Rewards.

#	Reward	Formula	Coefficient
1	$r_{position}$	$\frac{1}{T_r}\frac{1}{1+\lVert\vec{x}-\vec{r_{goal}}\rVert^2}$ if $t > T - T_r$	10.0
2	r_{pos_bias}	$\frac{\dot{\vec{x}}\cdot(\vec{r_{goal}}-\vec{x})}{\lVert\dot{\vec{x}}\rVert\lVert\vec{r_{goal}}-\vec{x}\rVert}$	1.0
3	r_{stall}	-1 if $\lVert\dot{\vec{x}}\rVert < 0.1m/s$ and $\lVert\vec{x}-\vec{r_{goal}}\rVert > 0.5m$	1.0
4	r_{face_goal}	$-\lVert\theta-\theta_{goal}\rVert$ if $\lVert\vec{x}-\vec{r_{goal}}\rVert > 0.5m$	0.1

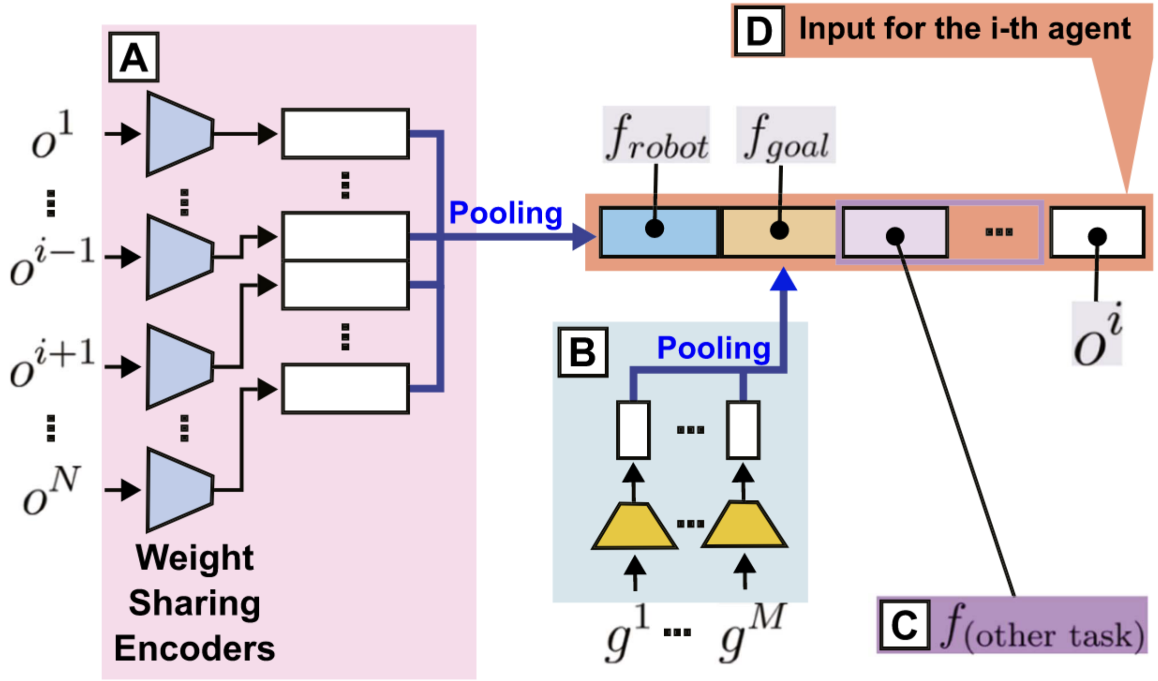


Fig. 2. Encoding of high-level policy input for the i-th robot. (A) all neighboring robot states are passed to a shared encoder and pooled to generate fixed size feature vector. (B) The same operation is done for the goal states, using different weights. (C) The feature space can be extended with new kinds of inputs (e.g., obstacles or opponents) depending on the task. (D) The concatenated vector of encoded features from (A, B, C) and the local observation O^i is the input to the high-level policy for the i-th robot.

TABLE I
REWARDS

MRMG Navigation

Reward	Description	Scale
Termination	1.0 if game success	10.0
Distance to Goal	$exp(-(\text{distance to unreached goals})^2)$	5.0
Motion bonus	clip(moving speed, 0.0, 1.0)	1.0
Neighbor distance	-1.0 if too close (<1 m)	1.0
Collision	-1.0 if robot collides	2.0

Box Packing

Reward	Description	Scale
Termination	1.0 if game success	5.0
Progress	$N_{\text{completed boxes}}/N_{\text{total box count}}$	0.25
Box velocity	$\sum_{\text{boxes}}(v_{\text{box}} \cdot \text{direction to goal})$	0.1
Box position	$\sum_{\text{boxes}} exp(-(p_{\text{box}} - p_{\text{goal}})^2)$	0.5
Neighbor distance	-1.0 if too close (<1 m)	1.0

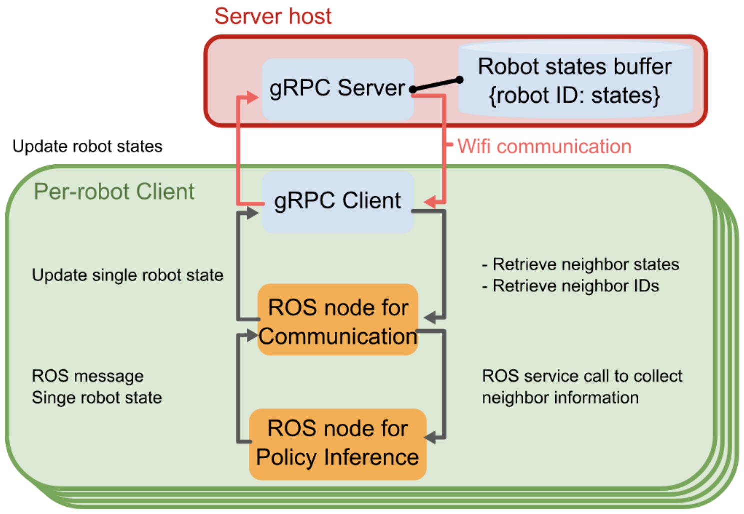


Fig. 4. gRPC communication framework. A gRPC server contains a message buffer with ID-state pairs. Each robot has a unique ID used for robot identification when communicating with clients.

Thank you