Integrating Convolutional Neural Networks and Reinforcement Learning for Autonomous Person-Following in Social Robotics

David Santiago Ortiz Almanza[1*]
and Fernando Enrique Lozano Martinez[1]

[1*]Department of Electrical and Electronic Engineering, University of the Andes, Bogotá, 111711, Bogotá, Colombia.

*Corresponding author(s). E-mail(s): ds.ortiz@uniandes.edu.co;
Contributing authors: flozano@uniandes.edu.co;

Abstract

The integration of advanced robotics with sophisticated machine learning techniques offers substantial opportunities to enhance the autonomous capabilities of social robots. This paper presents a novel approach that enables a Pepper-type social robot to autonomously follow a person. Our methodology combines Convolutional Neural Networks (CNNs) and Reinforcement Learning (RL) within a modular architecture. This architecture utilizes both 2D and 3D cameras as sensors, which provide robust real-time detection and tracking, while RL allows adaptive movement control. We have tested our implementation, demonstrating high performance and reliability across various complex environments. The results from these tests confirm that the proposed architecture and learning methods greatly enhance the robot's ability to autonomously follow a person. This research represents a significant advancement in the practical deployment of autonomous systems in the field of social robotics.

Keywords: Person Following, Convolutional Neural Networks (CNN), Reinforcement Learning (RL), Social Robotics

1 Introduction

The field of robotics has evolved significantly, transitioning from primarily automated industrial applications to include interactive roles in everyday human activities.

Within this domain, social robotics emerges as a critical area focused on developing robots capable of engaging in meaningful social interactions. These robots are not merely tools of efficiency but act as companions that enrich human life. In fact, de Graaf et al. (2015) characterize social robots as designed to emulate human emotions, engage in effective communication, and build relationships to enhance quality of life through personal interaction and support. Similarly, Hegel et al. (2009) highlight the ability of social robots to adapt to the emotional and social dynamics of human interactions, proposing their potential to provide companionship and assistance in daily activities.

An essential capability for broadening the utility of social robots in dynamic human environments is their autonomous ability to follow a person. This functionality could significantly benefit the elderly by providing continuous support and promptly responding to emergent situations, thus ensuring their safety (Tomoya et al., 2017). Furthermore, such capabilities could reduce burdens in environments where individuals frequently transport items, such as helping shoppers carry their purchases in supermarkets or managing luggage in airports (Ferreira et al., 2016), thereby allowing people to move more freely.

Autonomous person-following involves two critical technical components: the detection and tracking of the target individual, and the deployment of a reliable control system to maintain pursuit. Both components must integrate seamlessly; an advanced detection and tracking system is ineffective without a proficient control system to execute the pursuit, and vice versa. The challenges in detection and tracking, such as handling occlusions, variable lighting, and changes in the target's pose, necessitate high-performance solutions that may impact processing speed and real-time functionality (Chen et al., 2017). To address these requirements effectively, this paper proposes the integration of YOLO-NAS (Aharon et al., 2021), a convolutional neural network-based object detection model known for superior performance with minimal latency, and StrongSORT++ (Du et al., 2023), a state-of-the-art multi-object tracking method.

Moreover, the control system must rapidly process information from detection and tracking, making instantaneous decisions to adjust robotic parameters such as linear velocities (v_x, v_y), angular velocity (v_θ), and component movements, like the robot's head. It is crucial for the system to maintain an optimal distance from the person being followed while adeptly managing variations in the person's speed and other dynamic movements. This paper proposes a control guided by a policy learned through reinforcement learning, which adapts continuously to environmental interactions, optimizing behavior based on feedback. This is pivotal for adapting to dynamic changes in the followed person's behavior and is ideal for unstructured and evolving environments.

To support the autonomous following task, this paper introduces a modular architecture composed of four main modules: Detection and Tracking, 3D Pose Estimation,

Robot Control, and Recovery Behavior. This system has proven effective using only RGB and depth cameras as input sensors.

The primary contributions of this paper are: 1) A reusable, modular architecture that effectively equips a robot with autonomous following capabilities and can be extended with new modules to implement additional functionalities; and 2) A robot following behavior that efficiently integrates convolutional neural networks and reinforcement learning, capable of operating in complex scenarios.

2 Related Work

The literature presents various approaches to the autonomous person-following task, which includes diverse methods, processes, and algorithms for person detection, tracking, and subsequent following. Kawarazaki et al. (2015) employs a laser range scanner to detect people by identifying the shins of the target person under shin cluster conditions. This same sensor is used for obstacle detection. The control system implemented utilizes simple proportional control, where the speed is proportional to the distance from the person. However, the system may face limitations in outdoor environments or in scenarios where multiple people or obstacles interfere with the scanner, making it challenging to distinguish between multiple shins or similarly shaped objects.

Moreover, Awai et al. (2013) extends the use of a laser range finder by incorporating a camera as a sensor. This system combines features from Histograms of Oriented Gradients (HOG), color information, and shape data from the range to detect a person. However, the reliance on color-based features can be less robust in challenging situations such as lighting changes or alterations in the appearance of the person being followed, such as removing or donning a jacket. It also does not address potential cases of partial or total occlusions.

To address these adversities, Algabri and Choi (2020) integrates a color feature approach with a deep learning model. It utilizes the Single Shot MultiBox detector and Hue-Saturation-Value Histogram to identify and follow people in environments with multiple individuals and moderate lighting changes. Additionally, it employs SLAM for safe and autonomous navigation around obstacles.

Other alternatives have proposed using different types of sensors to gather depth information, such as stereo cameras. For instance, Chen et al. (2017) outlines the development of a person-following system for mobile robots using a modified algorithm called Selected Online Ada-Boosting (SOAB) that incorporates depth information from a stereo camera. This approach enables the robot to track a person in real-time, even in dynamic and complex environments, such as during partial or complete occlusions, changes in the person's appearance, or when multiple individuals wear similar clothing.

Regarding the use of reinforcement learning, Nguyen Ngoc et al. (2021) explores the development of a robotic assistant that follows people using visual target navigation and reinforcement learning techniques. The system integrates reinforcement learning with convolutional and recurrent neural network architectures, allowing the robot to effectively learn how to navigate and follow visual targets in a simulated environment. It employs an Advantage Actor-Critic (A2C) approach and is integrated into the ROS/Gazebo framework, enabling detailed simulation and efficient robot control.

3 Methodology

In this section, we describe the comprehensive methodology employed to develop and evaluate the autonomous person-following capabilities of the Pepper robot within the domain of social robotics. We detail the specific technologies, algorithms, and modular architecture utilized, explaining how each component integrates to facilitate effective robot operation. This includes a thorough discussion on both the simulated environment used for initial testing and the physical integration with the real robot using the Robot Operating System (ROS).

3.1 Pepper Robot

Pepper, a humanoid robot shown in Figure 1, was developed by Softbank Robotics in 2014, was designed to function as a social robot, capable of interacting, cohabitating, and potentially assisting humans. It is equipped with an array of sensors that facilitate natural interactions, such as loudspeakers and microphones for auditory communication, and RGB cameras along with a depth camera to perceive its surroundings. Additionally, an integrated tablet enhances its interactive capabilities with humans. Pepper also incorporates various sensors including an inertial unit, lasers, infrared, sonars, Motion Responsive Elements (MRE), and contact and tactile sensors, which collectively enable it to understand environmental dynamics. The robot's mobility and manipulation are powered by 14 motors, allowing movement of its head, shoulders, elbows, wrists, hands, hips, knees, and enabling 2D spatial navigation.

These features have established Pepper as a widely used platform in robotics research, covering areas such as autonomous navigation (Garrido Urbano, 2020; González Gutiérrez, 2022), emotion detection (Padilla Torres, 2023), the implementation of large language models for human interaction (Romero Colmenares and Rojas Becerra, 2023), and human entertainment through reinforcement learning (García Cárdenas, 2019), among others. In this study, Pepper will be employed in both simulated and physical environments to perform autonomous person-following tasks. However, the research is designed to be applicable and extendable to any robotic platform equipped with at least a 2D camera, a depth camera, and the capability for spatial movement.

Fig. 1: Pepper Robot

3.2 Person-Following Modular Architecture

Figure 2 presents the diagram of the proposed modular architecture designed for the autonomous person-following task. This architecture comprises four main modules: Detection and Tracking, 3D Pose Estimation, Robot Control, and Recovery Behavior, providing an end-to-end integration from the initial perception inputs captured by the robot's cameras to the control of its movements. Next, we describe briefly each of these modules:

- **Detection and Tracking Module:** This module is responsible for identifying the target person and maintaining robust, real-time tracking under various challenges such as poor lighting, partial or complete occlusions, and differentiating individuals in crowded environments. It processes raw 2D images captured by the Pepper robot's camera top, outputting the x, y coordinates of the center of the target person's bounding box in the original image.

- **3D Pose Estimation Module:** Following the detection stage, this module computes the relative 3D coordinates of the tracked person concerning the robot, using the bounding box coordinates provided by the previous module. It utilizes the robot's 3D sensor to determine the distance to the person. The output of this module includes the x, y, z coordinates of the person being followed.

- **Robot Control Module:** Tasked with motion control, this module sends instructions to adjust the robot's linear and angular velocities and the head joints based on the relative position obtained from the 3D Pose Estimation module.

- **Recovery Behavior Module:** This module addresses situations where the person has moved out of the robot's visual range and tracking is lost. It implements high-level actions to reacquire the target. Two strategies are used: first, the robot rotates about its axis to search for the person; if unsuccessful, the robot employs its social capabilities to verbally inform the person that they have been lost and to request their return to the visual field to resume tracking.

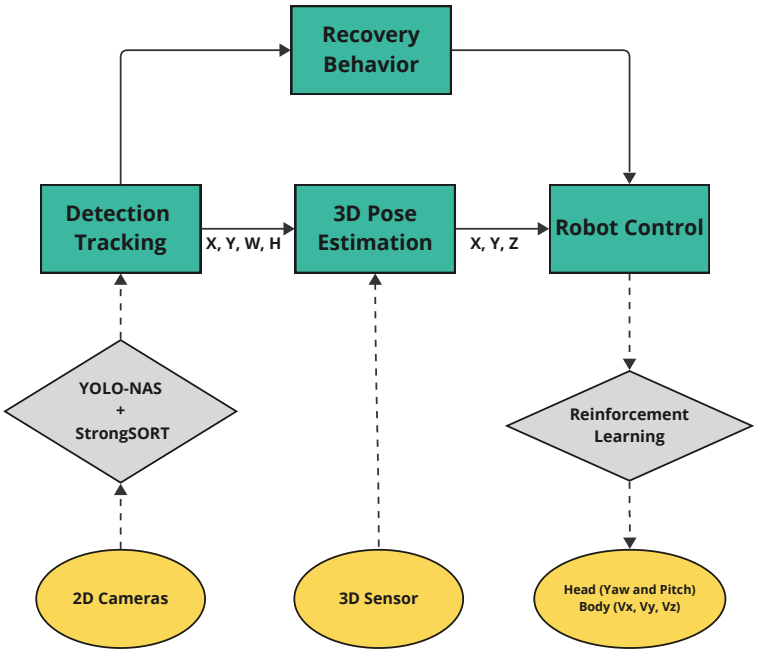


Fig. 2: Modular architecture for autonomous person-following. The architecture is structured around four main modules depicted as green boxes: Detection and Tracking, 3D Pose Estimation, Robot Control, and Recovery Behavior. Additionally, yellow ovals represent the robot's sensors or actuators involved, while gray diamonds denote the underlying technologies to be utilized.

3.2.1 Detection and Tracking

As previously mentioned, the Detection and Tracking Module is pivotal for robustly identifying the target person and maintaining real-time tracking from the raw 2D images provided by the robot's camera. To meet these requirements and address the associated challenges, an integration of YOLO-NAS (Aharon et al., 2021) and StrongSORT++ (Du et al., 2023) has been implemented.

The YOLO (You Only Look Once) framework was chosen as the person detection model due to its exceptional capability to balance speed and accuracy. This is achieved through its unique approach of applying a single neural network to the entire image, thereby predicting bounding boxes and class probabilities in one evaluation (Redmon et al., 2016). Specifically, YOLO-NAS (Aharon et al., 2021), a cutting-edge foundational object detection model based on Neural Architecture Search, was selected. It addresses significant challenges such as detecting small objects, improving localization accuracy, and enhancing the performance-per-compute ratio. These features are crucial for dynamic environments like autonomous person-following where real-time processing is essential. YOLO-NAS introduces several innovative elements including quantization-aware modules, automatic architecture design via AutoNAC, and a hybrid quantization method that selectively quantizes parts of the model to optimize both latency and accuracy (Terven et al., 2023).

Figure 3 illustrates the Accuracy (mAP) versus Latency (ms) graph measured on the COCO dataset using NVIDIA T4, showcasing that YOLO-NAS achieves comparable or superior accuracy to other state-of-the-art models like YOLO-V7 and YOLO-V8, but with significantly reduced inference time, making it an ideal candidate for this task.

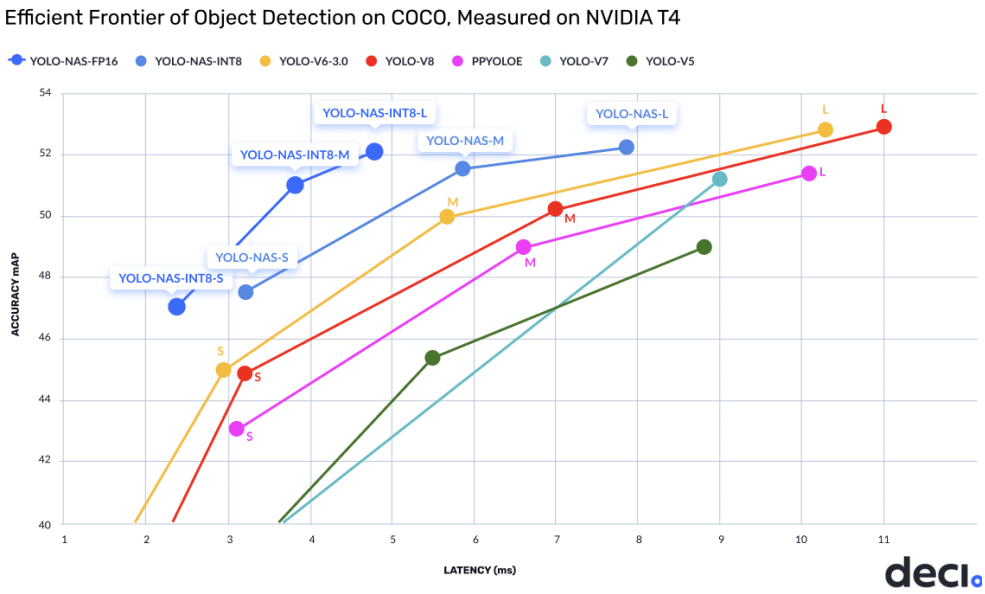


Fig. 3: Efficiency Frontier plot for object detection on the COCO dataset. Image taken from Deci-AI super-gradients GitHub (Aharon et al., 2021)

However, while YOLO excels in detecting objects independently in each video frame, it lacks the capability to track an object over time. To overcome this limitation, we integrated StrongSORT++, an advanced version of the multi-object tracking (MOT) algorithm DeepSORT. StrongSORT++ enhances classic tracking capabilities by addressing two main challenges in MOT: missing associations and missing detections. It incorporates two lightweight, plug-and-play algorithms: the Appearance-Free Link Model (AFLink) and Gaussian-Smoothed Interpolation (GSI) (Du et al., 2023). This combination significantly improves multi-object tracking accuracy, particularly in environments with high occlusion and crowded scenes as demonstrated on various benchmarks including MOT17, MOT20, DanceTrack, and KITTI. Figure 4 shows IDF1-MOTA-HOTA of SOTA trackers on MOT17 and MOT20 test sets.

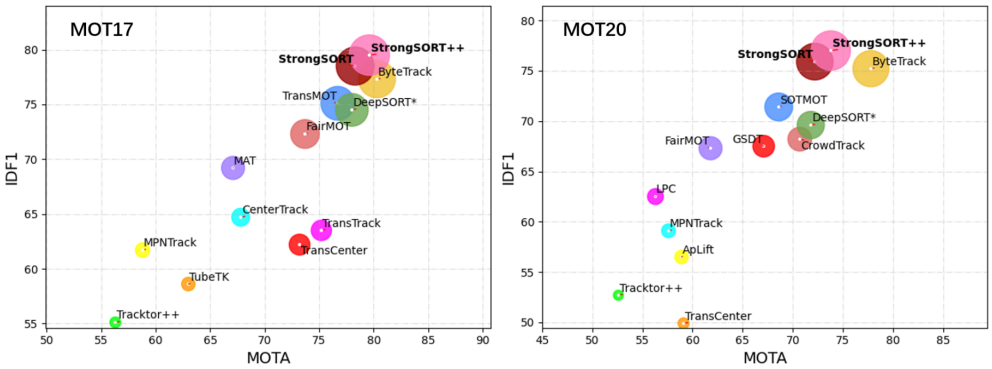


Fig. 4: IDF1-MOTA-HOTA of SOTA trackers on MOT17 and MOT20 test sets. Image taken from StrongSORT official GitHub (Du et al., 2023)

The integration of YOLO-NAS and StrongSORT++ provides a robust and efficient detection and tracking model that operates effectively even in complex scenarios, and is capable of real-time performance. This ensures seamless end-to-end integration from visual perception to motion control in the modular architecture of autonomous person-following in social robotics.

3.2.2 3D Pose Estimation

Following the successful identification and tracking of the target person, the 3D Pose Estimation Module calculates the relative three-dimensional coordinates of the tracked person with respect to the robot. This computation leverages the bounding box coordinates (x, y, width w, and height h) provided by the Detection and Tracking Module. Utilizing the robot's 3D sensor, this module determines the distance to the person and outputs the coordinates (x, y, z) of the person being followed.

Although the Detection and Tracking Module provides the 2D coordinates from the raw image captured by the RGB camera, these alone are insufficient for dynamic person-following. Decisions such as adjusting speed, stopping, or reversing require

knowledge of the person's distance from the robot. Ideally, the robot should maintain a fixed safety distance from the person. Hence, the 3D Pose Estimation Module plays a critical role in ensuring the robot can navigate effectively while keeping this safety distance.

Implementing this module is not straightforward due to the physical separation and differing perspectives of the RGB and 3D sensors on the robot. For instance, in the Pepper robot, the RGB camera is located at the front while the 3D sensor is positioned independently at the left eye. Figure 5 illustrates the placement of these cameras, highlighting the challenges of inaccurate associations due to their different locations, viewpoints, and viewing angles. This disparity necessitates a robust calibration process between the depth camera and the RGB camera to ensure precise distance measurements.

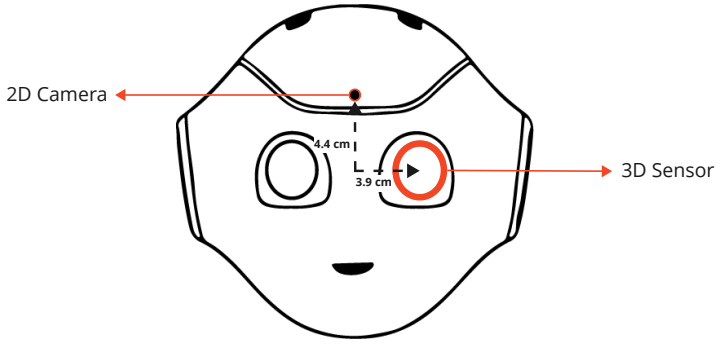


Fig. 5: Pepper 2D camera and 3D sensor positions

Singh et al. (2020) offers a comprehensive analysis of various calibration techniques. It compares methods such as homography, the fundamental matrix, and proposed translation estimation techniques to address the alignment issue. Metrics such as root mean square error (RMSE) and error variance are employed to assess the effectiveness of these methods. Ultimately, it concludes that utilizing a homogeneous transformation matrix H, estimated using homography provides the most accurate results. Equation 3 details the homography matrix estimated by this study.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

$$H = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$H = \begin{bmatrix} 8.7795 \times 10^{-1} & 2.7578 \times 10^{-2} & 2.8110 \times 10^{1} \\ 2.4932 \times 10^{-3} & 8.5721 \times 10^{-1} & -6.3316 \times 10^{0} \\ -1.9371 \times 10^{-4} & 4.2107 \times 10^{-5} & 1.0000 \times 10^{0} \end{bmatrix} \tag{3}$$

These findings have been adopted as the calibration method of choice in this paper, ensuring that the 3D Pose Estimation Module can reliably compute the distance of the person from the robot, which is pivotal for the autonomous person-following task in dynamic and potentially crowded environments.

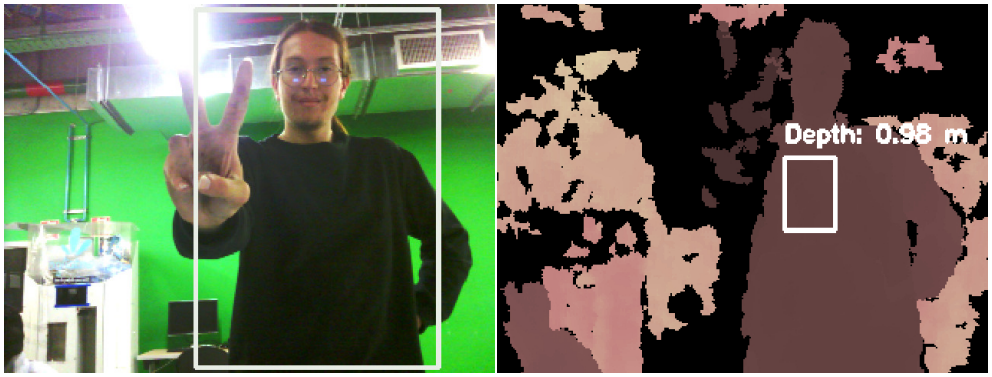


Fig. 6: Visualization of the distance estimation process. Left: 2D camera view with bounding box generated by the detection and tracking module. Right: Aligned 3D sensor view with final bounding box and calculated distance.

After applying the calibration method described to align the 2D camera with Pepper's 3D sensor, the next step involved determining the effective distance from the robot to the person. To achieve this, we utilized the bounding box provided by the detection and tracking module, trimming it by 80% in both width and height dimensions. This adjustment aimed to create a smaller bounding box centered on the person, thereby minimizing the inclusion of noisy pixels that could distort distance estimation due to their proximity or distance from the sensor.

Subsequently, we computed the median distance of the pixels within this final bounding box. The use of the median, a robust statistical measure less sensitive to outliers that might still be present within the box, ensured a more reliable estimation of the person's distance from Pepper. Figure 6 illustrates the process: the left image shows the 2D camera view with the bounding box generated by the detection and tracking module, while the right image displays the 3D sensor view post-alignment, featuring the final bounding box and the calculated distance.

3.2.3 Robot Control

The Robot Control Module is crucial for managing the motion of the robot based on inputs from the Detection and Tracking and the 3D Pose Estimation Modules. It

adjusts the robot's linear and angular velocities and the head joints according to the relative position of the person being tracked. This control mechanism is underpinned by principles of reinforcement learning (RL), where an agent learns optimal behaviors through interactions within its environment.

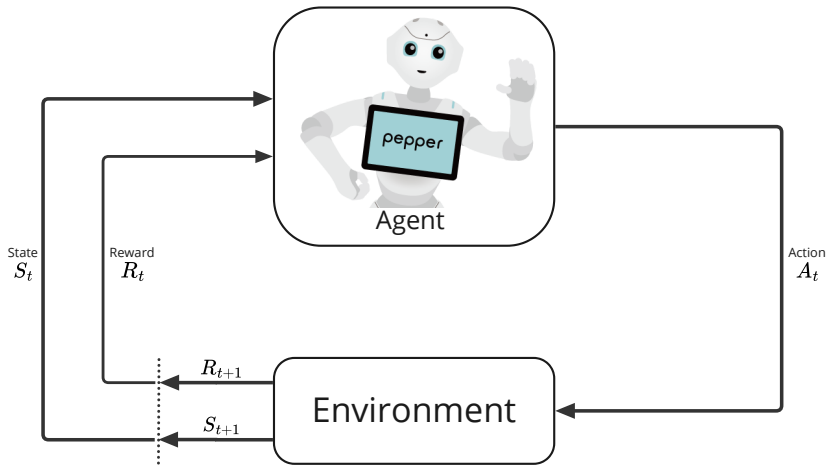


Fig. 7: Agent-Environment Interaction in a MDP. Image Inspired From "Reinforcement Learning: An Introduction" (Sutton and Barto, 2018)

Reinforcement learning is fundamentally a trial-and-error learning process represented mathematically as Markov Decision Processes (MDPs). Within this framework, an agent observes a state of the world s_t, selects an action a_t, receives a reward r_t from the environment, and aims to maximize its cumulative reward. The interaction between the agent and the environment in a Markov decision process is depicted in Figure 7. The primary goal is to develop a policy—a set of rules that dictate the agent's actions based on current states.

RL algorithms are typically divided into two categories: model-based and model-free. Model-based methods construct an internal model of the environment and use it for planning, while model-free methods learn directly from the interaction with the environment without constructing a model. Given the uncertainties in real-world environments such as autonomous navigation in a person-following task, model-free methods are advantageous due to their robustness and ease of implementation. These methods directly learn either the values of actions taken in states or the policies themselves, enabling optimal behavior derivation without the need to simulate environmental models (Dayan and Niv, 2008).

Within the domain of model-free RL, there are two primary families: Policy Optimization and Q-learning. Policy Optimization methods, such as Proximal Policy Optimization (PPO), directly adjust the policy function, often leading to stable and consistent performance. Conversely, Q-learning methods work by updating a Q-function that estimates the value of actions, which can be more sample efficient but potentially less stable (OpenAI, 2018b).

The Robot Control Module employs Proximal Policy Optimization (PPO) (Schulman et al., 2017) due to its balance of ease of implementation, stability, and performance in continuous control tasks. PPO has shown to be particularly suited to environments with high variability and changing conditions similar to this one, such as autonomous navigation (Taheri and Hosseini, 2024) and manipulation (Yang et al., 2024). It utilizes a clipping function in the objective to prevent drastic policy updates, maintaining close proximity to the previous policy. This is achieved by defining a ratio $r_t(\theta)$ between the probabilities of the new and old policies for a given action, and applying a clipped objective $L^{CLIP}(\theta)$ that limits the update ratio within a specified range (ϵ). The main objective is show in Equation 4. This approach reduces the risk of detrimental large updates, enhancing both the stability and efficiency of policy gradient methods (OpenAI, 2018a).

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \right] \tag{4}$$

3.2.3.1 RL environment definition

The reinforcement learning environment for the autonomous person-following task is defined by its states, actions, and reward structure, which are tailored to reflect the dynamic interactions between the robot and the person it is tracking.

1. **State Space**

   The state space is an 8-dimensional vector represented as:

   $$s_t = (c_x, c_y, d, v_x, v_y, v_z, yaw, pitch) \tag{5}$$

   where:

   - c_x, c_y: Coordinates of the person being followed in the robot's frame of reference.
   - d: Distance from the robot to the person.
   - v_x, v_y: Current linear velocities of the robot along the x and y axes, respectively.
   - v_z: Current angular velocity of the robot.
   - $yaw, pitch$: Current yaw and pitch angles of the robot's head.

   These dimensions provide a comprehensive description of both the relative position and orientation of the person with respect to the robot, as well as the robot's own state of motion.

2. **Action Space**

The action space consists of a 5-dimensional vector

$$a_t = (\delta v_x, \delta v_y, \delta v_z, \delta yaw, \delta pitch) \tag{6}$$

where each component represents a change to be applied to the corresponding state dimension. This delta-based action design was chosen following extensive experimentation, which showed that incremental changes lead to smoother, more naturally coordinated movements, enhancing the robot's ability to maintain a stable and fluid following behavior

3. **Reward Function**

The reward function is designed to closely align with the primary goal of maintaining an optimal following distance while ensuring safe interaction. It is defined as:

$$R(s_t, a_t) = 1 - |d - d_{desired}| \tag{7}$$

where d is the current distance from the robot to the person and $d_{desired}$ is the desired distance to keep, 1 meter for this case. The reward is inversely related to the difference with the desired distance, encouraging the robot to minimize it within a safe range. If the distance falls below 0.5 meters (indicating too close proximity) or exceeds 2 meters (indicating loss of tracking), the episode is terminated and a reward of -2000 is given. This design encourages the robot to maintain a desirable distance without biasing its behavior towards other unintended actions.

3.2.4 Recovery Behavior

The Recovery Behavior Module serves as an emergency response mechanism to address instances where the target person escapes the visual field of the robot for a long while, resulting in a loss of tracking. This module is designed to autonomously re-establish tracking with minimal disruption.

In the event that the Detection and Tracking Module—utilizing StrongSORT++ for tracking—fails to maintain visual contact with the target for a predetermined interval (defaulted to three seconds but adjustable), the robot suspends its directed following behavior. The initial response involves the robot pausing and then commencing a rotational search on its axis, aimed at rediscovering the target within its visual perimeter. This rotational movement is designed to maximize the likelihood of reacquiring a visual lock on the target, allowing the Detection and Tracking Module to recognize and resume tracking the same individual.

Should this initial search prove fruitless—either through failure to visually relocate the person or the tracker's inability to re-associate the target as the same entity—the module progresses to its secondary protocol. This involves activating the robot's speech capabilities to engage socially with the environment. The robot issues

a verbal notification expressing that it has lost sight of the person and requests their assistance to return to the visual field. It instructs the person to position themselves directly in front of it and to physically interact by pressing its head, a gesture that resets the tracking protocol. The robot will then identify the most centrally located individual as the new tracking target.

Although resorting to this secondary strategy is less ideal, it is a necessary provision to ensure continuity in tracking operations when visual methods fail. This dual-strategy framework equips the robot with the necessary tools to manage disruptions in tracking through a combination of technical maneuvers and social interactions, ensuring robustness and adaptability in dynamic environments.

3.3 Simulation Environment: qiBullet

qiBullet (Busy and Caniot, 2019) is an open-source simulation tool developed by SoftBank Robotics. It utilizes the Bullet physics (Coumans and Bai, 2016–2021) engine to emulate the Pepper and NAO robots, commonly employed in research and educational settings. The simulator provides a Python API and ROS framework compatibility, enabling interactions with virtual robot models in complex environments. This facilitates machine learning tasks and testing of various scenarios.

When integrating the detection and tracking module with YOLO-NAS and StrongSORT++ by sending images individually, a significant performance issue in inference time was encountered. This observation was in stark contrast to earlier experiments conducted with other videos and webcams on the same hardware. Analysis revealed that these models were optimized for processing a continuous stream of images, and their efficiency decreased notably when handling images one by one. To overcome this, a method involving concurrent communication through sockets and threads was developed. Images are now transmitted continuously via a socket on a dedicated thread, while the detection module receives an uninterrupted stream of images through this socket. This approach led to a 45-fold improvement in inference time. Figure 8 illustrates the diagram of the socket stream between the client (Detection and Tracking Module) and the server (Simulation GUI). Table 1 presents a comparison of the performance metrics before and after this adjustment.

Metric	Before Optimization	After Optimization	Improvement Factor
Average Inference Time (ms)	131.2	2.71	48.41
Average Post-processing Time (ms)	31.4	0.62	50.64
Frames Processed Per Second (FPS)	6.15	300.30	48.83

Table 1: Comparative Performance Metrics of Detection and Tracking Module Before and After Implementing Socket-based Continuous Image Streaming with qiBullet GUI.

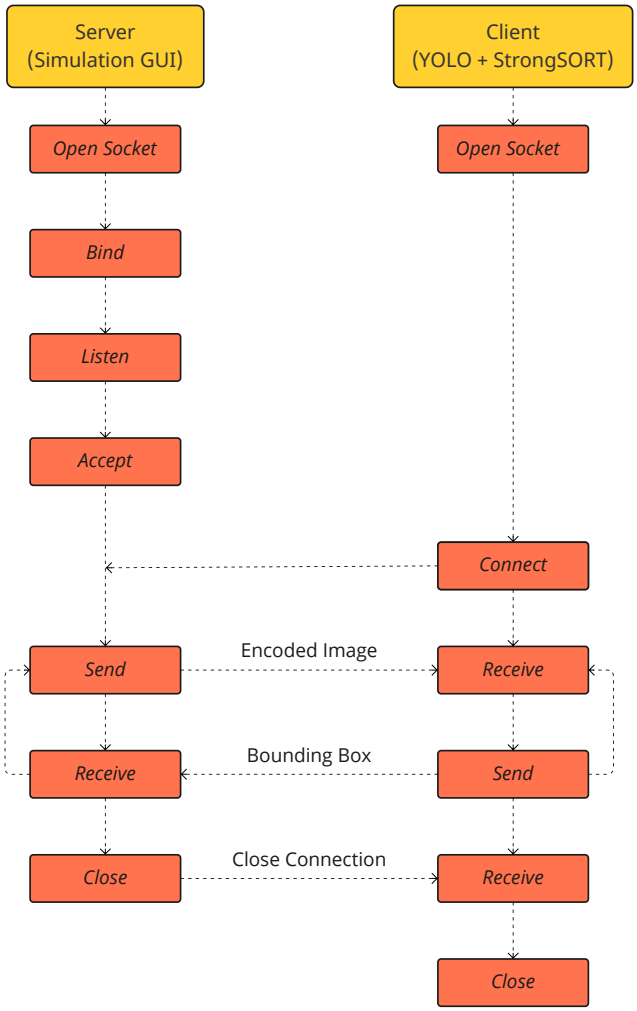
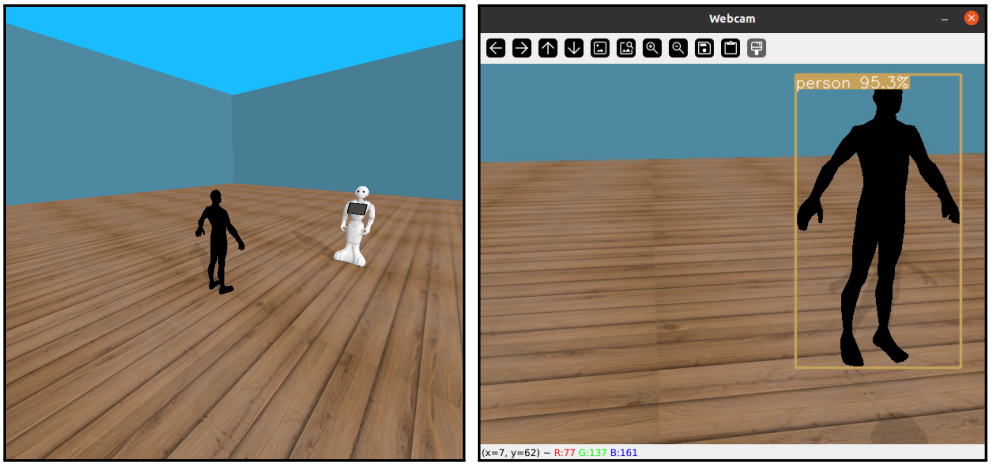


Fig. 8: Schematic representation of the continuous image stream transmission via socket communication between the client (Detection and Tracking Module) and the server (Simulation GUI), demonstrating the setup that achieved a significant reduction in inference time.

After the proper integration of the detection and tracking module with the 3D Pose Estimation module into the qiBullet simulator. A simulation environment featuring a human model was constructed to facilitate this integration. Figure 9 shows the qiBullet environment along with the detection and tracking module using the 2D camera of the Pepper robot.

The human model was endowed with the capability to move randomly through the environment, where it continuously made one of the following three decisions randomly: move a random distance, rotate around its axis by a random number of degrees, or stop for a random duration. These three actions were designed to simulate the way a person navigates through an environment as realistically as possible. Moreover, the implementation of random actions with random values ensures that the reinforcement learning model can generalize the task of following a person, rather than merely overfitting to a single trajectory.



(a) qiBullet Simulation Environment **(b)** Pepper POV with YOLO and StrongSORT++

Fig. 9: Simulation Environment: qiBullet

3.4 Physical Integration: ROS Architecture

To integrate the previously proposed modular architecture with a Pepper robot, the Robot Operating System (ROS) must be employed as the framework. The use of ROS is crucial due to the limited computational capacity of Pepper robots. Although these robots are equipped with numerous sensors and actuators, they lack sufficient computational power to handle complex processes internally. ROS enables a distributed architecture where data from the Pepper robot is transmitted to a local computer, which possesses significant processing capabilities. This computer then processes the incoming data through the person-following modular architecture and calculates the necessary actions for the robot to execute. These actions are subsequently relayed back to the robot via ROS. Communication within ROS primarily utilizes topics for asynchronous and unidirectional communication, and services for bidirectional, synchronous communication, with WiFi as the communication medium. This configuration facilitates the decoupling of processing tasks, thus allowing real-time operations. Figure 10 illustrates how the modular architecture is integrated with ROS.

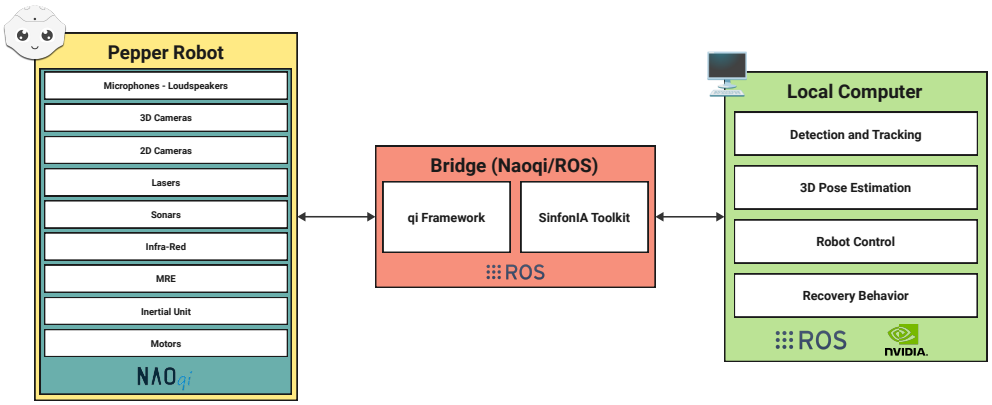


Fig. 10: Integration of Modular Architecture with Robot Operating System (ROS)

4 Results and Discussion

In this section, we present and discuss the results of our study on the autonomous person-following capabilities of the Pepper robot. We critically evaluate the performance of each module (detection and tracking, 3D pose estimation, robot control, and recovery behavior) as well as the overall system integration. This evaluation encompasses both simulated environments and physical implementations with the real robot using the Robot Operating System (ROS). Through comprehensive quantitative and qualitative analyses, we aim to substantiate the effectiveness of the integrated system and delve into the practical implications of our methodologies in realistic settings.

4.1 Detection and Tracking

While the quantitative assessment of integrating YOLO-NAS as the object detection model along with StrongSORT++ as the tracking model falls outside the scope of this research—since these models have been evaluated in their respective original publications (Aharon et al., 2021; Du et al., 2023)—this section will discuss the reported quantitative metrics and evaluate the local processing time performance to determine this module's capability to operate in real-time, as previously discussed.

4.1.1 YOLO-NAS Object Detection Model

YOLO-NAS offers six pretrained models on prominent datasets such as COCO, Objects365, and Roboflow 100. It provides three sizes (in terms of the number of parameters): small, medium, and large, with each size available in a standard and an INT-8 quantized version. Table 2 compares these models in terms of parameter count, mAP, and latency as reported in Aharon et al. (2021).

For this study, the medium-sized YOLO-NAS model quantized to INT-8 was selected due to its optimal balance between latency and mAP. As shown in the Efficiency Frontier plot for object detection on the COCO dataset (Figure 3), this version

Modelo	Number of Parameters (in Millions)	mAP	Latencia (ms)
YOLO-NAS S	22.2	47.5	3.21
YOLO-NAS M	58.2	51.55	5.85
YOLO-NAS L	79.4	52.22	7.87
YOLO-NAS S INT-8	22.2	47.03	2.36
YOLO-NAS M INT-8	58.2	51.0	3.78
YOLO-NAS L INT-8	79.4	52.1	4.78

Table 2: Comparative Analysis of YOLO-NAS Models. Table taken from Ultralytics YOLO-NAS documentation (Ultralytics, 2023)

exhibits significantly higher mAP compared to even the larger versions of other models, while maintaining a latency comparable to their small versions. This choice enabled the use of a high-performing model in real-time applications.

4.1.2 StrongSORT++ Tracking Performance

Figure 4 illustrates the quantitative results achieved by StrongSORT++ in terms of IDF1, MOTA, and HOTA metrics, as compared with other state-of-the-art (SOTA) trackers on MOT17 and MOT20 datasets. StrongSORT++ excels in all three metrics, achieving an IDF1 (ID F1 Score) of 79.5, which evaluates the accuracy and recall of correctly identified objects, indicating its effectiveness in maintaining consistent object identities across frames. The MOTA (Multiple Object Tracking Accuracy), scored at 79.6, measures overall tracking accuracy based on false positives, false negatives, and mistaken identities, demonstrating high performance with minimal error. Overall, StrongSORT++ proves to be an efficient and robust tracking algorithm, particularly in maintaining consistent and accurate object identities, making it a solid solution for specific applications that require precise and reliable tracking.

4.1.3 Inference Time Evaluation

A review of the inference time results for the combined detection and tracking module, which incorporates YOLO-NAS-M INT-8 with StrongSORT++, is presented in Table 3. These results were obtained using an AMD Ryzen 5 7600 6-Core Processor and NVIDIA RTX 4060TI 16GB. Analysis of the table shows that the majority of the processing time is attributed to StrongSORT++ (86.76%), while the inference time for YOLO-NAS is remarkably low (4 ms). However, the complete integration of these two technologies allows for processing an average of 30.48 FPS, which is ideal for our use case since the 2D camera of Pepper operates at a maximum of 30 FPS. In conclusion, the integration of YOLO-NAS-M and StrongSORT for detection and tracking a person provides robust performance, even in challenging conditions such as occlusions, lighting changes, and pose variations, crucially, in real-time.

Process	Average Inference Time (ms)	Average FPS Processed
YOLO-NAS-M INT-8	4.34	230.41
StrongSORT++	28.46	35.13
Complete Integration	32.8	30.48

Table 3: Inference time and FPS processed for YOLO-NAS-M INT-8, StrongSORT++, and their complete integration on AMD Ryzen 5 7600 6-Core Processor and NVIDIA RTX 4060TI 16GB.

4.2 3D Pose Estimation

To assess the performance of the implemented 3D pose estimation module, specifically its accuracy in determining the distance to the person being followed relative to the robot, a comprehensive evaluation was conducted. This evaluation utilized both the calibration method of Pepper's 3D sensor in relation to its 2D camera and the bounding box transformation, along with the median distance calculation from these pixels as detailed in the Methodology section 3.2.2. The assessment involved comparing the estimated distances from the module with actual physical measurements. For this, a sample of 25 records was collected, involving various individuals standing at different distances and angles from the robot. The discrepancies between the module's reported distances and the actual measurements were analyzed to calculate the estimation error. Figure 11 presents a boxplot illustrating the distribution of these estimation errors.

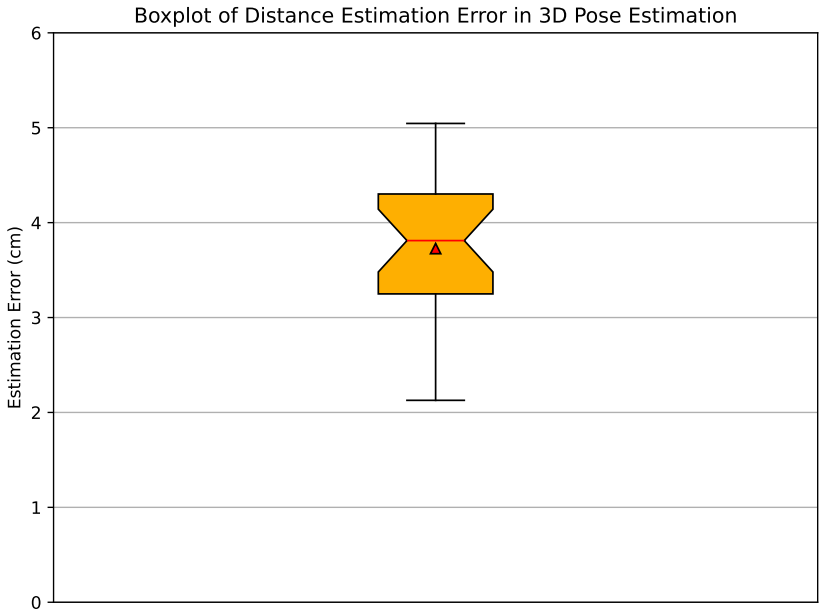


Fig. 11: Boxplot of Distance Estimation Error in 3D Pose Estimation

Upon analyzing the results depicted in Figure 11, it was observed that the module achieved an average estimation error of 3.73 cm, with a standard deviation of 0.67 cm. This error is considerably low, especially given the focus range of the 3D sensor, which spans from 40 cm to 8 meters. Notably, the minimum distance at which the 3D sensor can accurately determine a distance is 40 cm; below this threshold, Pepper is effectively "blind". Consequently, the minimum safe following distance was set at 50 cm to accommodate the estimation error, a distance also deemed sufficient for safety considerations. However, the ideal following distance for Pepper was established at 1 meter—twice the safety distance—with the maximum effective following distance capped at 2 meters, four times the safety distance. Beyond this range, effective following is considered excessively challenging. These distances were used as parameters for training the control module.

4.3 Robot Control

For the Robot Control module, we adhered to the methodology outlined in Section 3.2.3. The Proximal Policy Optimization (PPO) algorithm was trained over 3.2 million steps across three concurrent simulations, each running on an independent thread. We utilized the Reinforcement Learning (RL) environment specified and employed qiBullet as the simulation platform, as detailed in Section 3.3.

Figure 12 illustrates the moving average (window of 100 episodes) of accumulated rewards, averaged across the three simulations over the training episodes. The moving average was calculated to smooth the graph and better analyze the trend. During the initial 1000 episodes, the average accumulated reward was approximately -2000, indicating that the episodes tended to terminate quickly, resulting in the negative reward associated with episode failure. However, a clear trend of increasing average accumulated rewards is observable as training progresses. Around episode 3200, the average accumulated reward reaches equilibrium at approximately 0. This suggests that although episodes were still terminating, they lasted longer, allowing the agent to accumulate enough positive rewards to offset the negative reward from unsuccessful terminations. Towards the end of the training, near episode 5000, the average accumulated reward approaches 2500, the maximum configured reward for a successfully completed episode. This indicates that the agent had almost fully mastered the given environment, consistently achieving the maximum reward possible.

To evaluate the effectiveness of the developed policy in autonomously tracking a human subject, we designed three increasingly complex scenarios. In each scenario, the robot was tasked to follow a person along a predefined path from one end of the scenario to the other and back. Figure 13 displays images of these three scenarios both in the qiBullet environment and their corresponding 2D map representations, highlighting the origin and destination points and the anticipated human trajectory.

These scenarios are crucial for assessing person-following capabilities under varied conditions. Scenario 1 features a simple, linear path with a right-angle turn, serving as a basic test for the robot's tracking abilities in a controlled setting. Scenario 2 presents

multiple right-angle turns and a zigzag path, challenging the robot with frequent and sharp direction changes typical of environments like warehouses or crowded urban areas. Scenario 3 integrates elements from the previous settings but offering a more realistic and complex layout that simulates real-world settings such a real home. This scenario is ideal for testing the robot's capabilities to autonomously follow a person in intricate environments. Each scenario progressively introduces higher complexity and diverse routing challenges, allowing a thorough assessment of the robot's performance across different environments to ensure robustness and adaptability prior to real-world application.

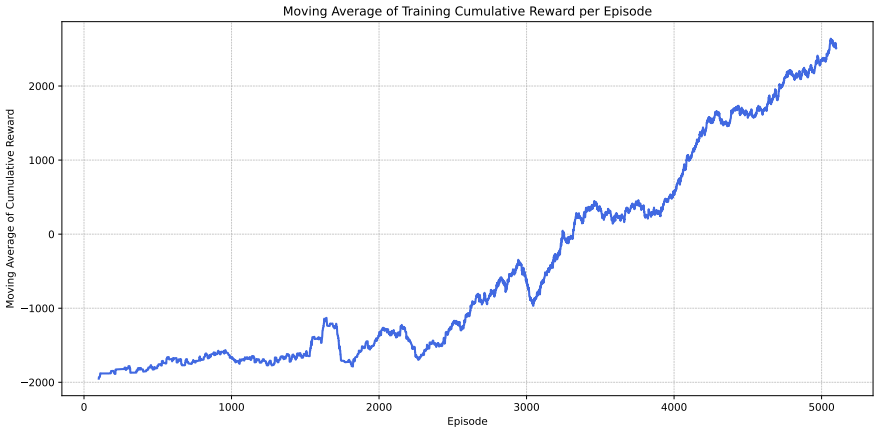
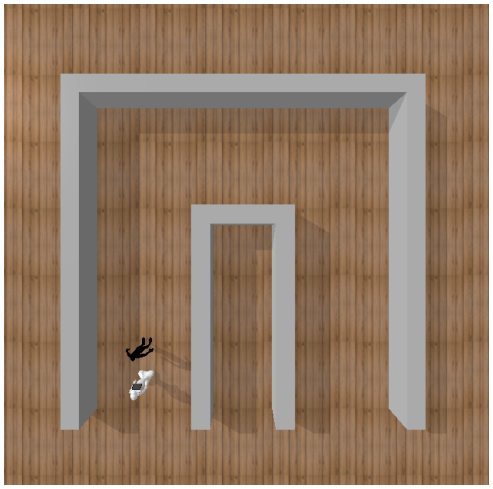


Fig. 12: Moving Average of Training Cumulative Reward per Episode

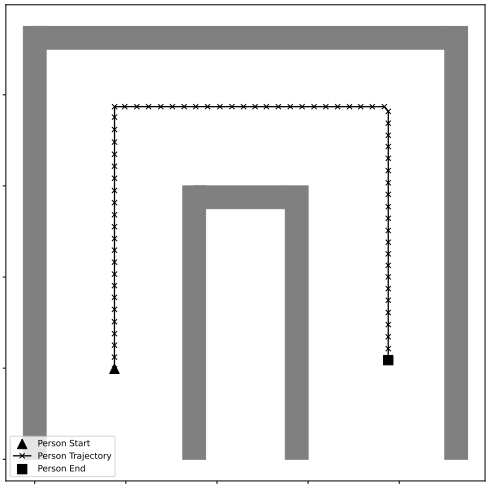
The outcomes from these scenarios revealed that the Pepper robot was capable of autonomously following the person in all three scenarios, maintaining a distance of 0.5 m to 2 m. Figure 14 displays the results, showing the trajectories for both the outbound (depicted with blue lines) and return (depicted with red lines) journeys. The gray line connecting pairs of points on the person's and the robot's trajectories indicates that these points correspond to the same moment in time.

Contrary to expectations, the robot did not precisely mimic the human's path. However, it maintained a similar trajectory that allowed it to keep the person within its field of view and adequately tracked throughout the entire course. Across all scenarios, the robot adeptly navigated straight paths, sharp left and right turns, and curvy tracks. It even learned to retreat and give space to the person if they turned towards the robot, as shown in the return trajectories. Videos associated with each scenario are available in the footnote. These results not only demonstrate that the reinforcement learning-based policy was robust enough to effectively perform the task of autonomously tracking a person but also that the overall modular architecture equipped the robot with the necessary capabilities to execute this task.

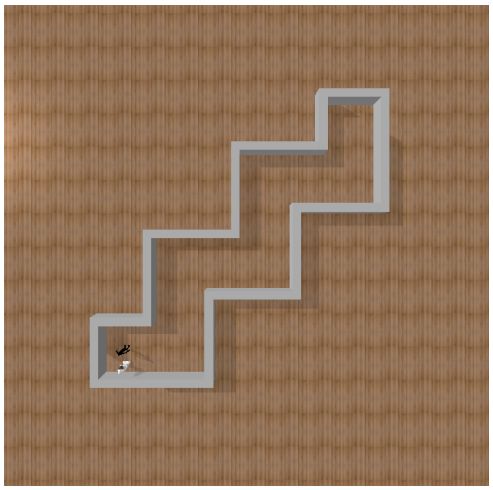
Video of Scenario 1
Video of Scenario 2
Video of Scenario 3



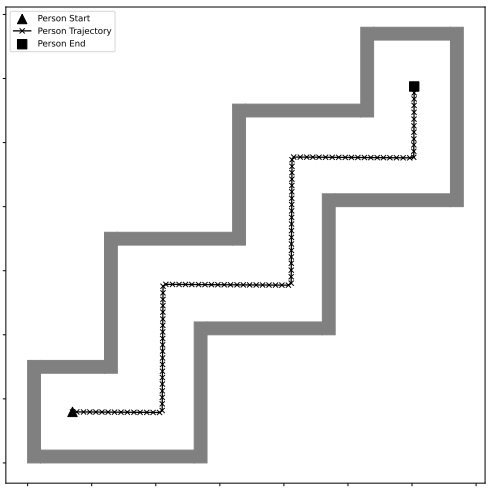
(a) Scenario 1 - qiBullet



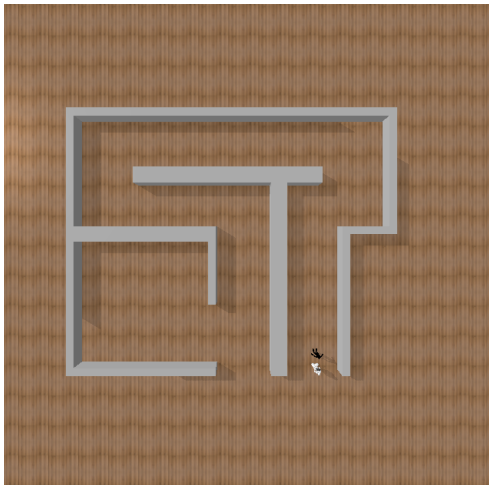
(b) Scenario 1 - Trajectory



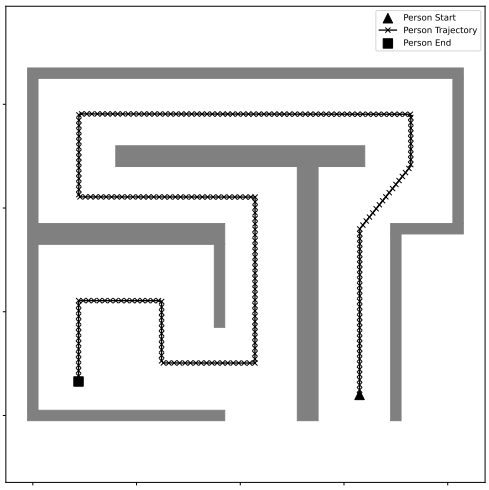
(c) Scenario 2 - qiBullet



(d) Scenario 2 - Trajectory

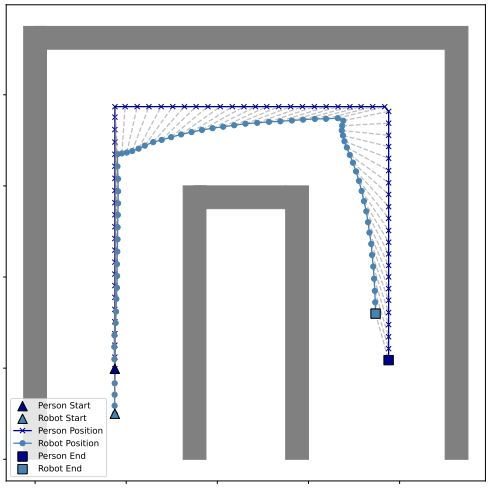


(e) Scenario 3 - qiBullet

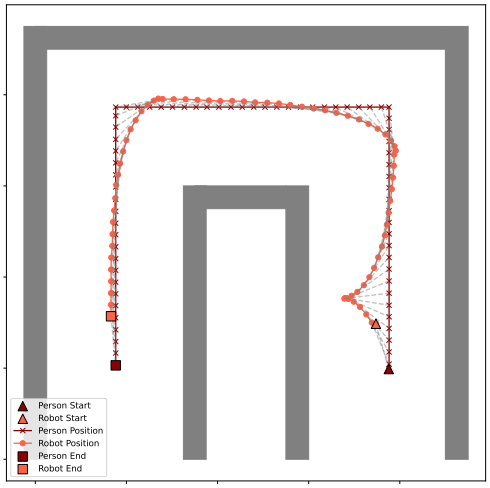


(f) Scenario 3 - Trajectory

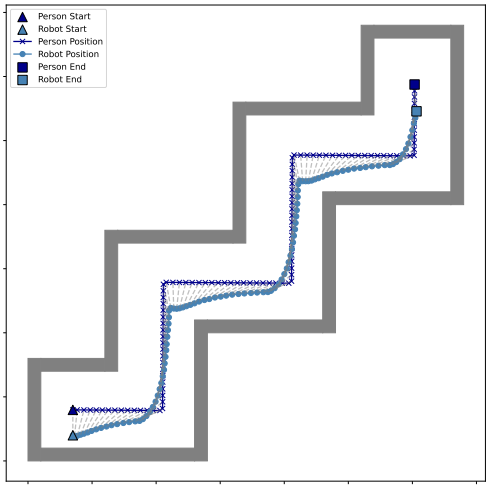
Fig. 13: Proposed Simulated Scenarios in qiBullet and Person Trajectory for Policy Evaluation



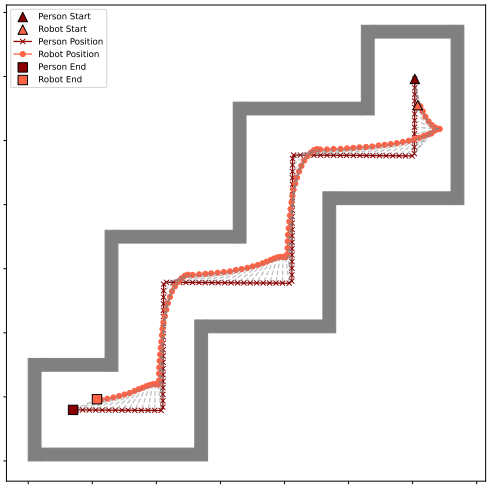
(a) Scenario 1 - Outbound Trajectory



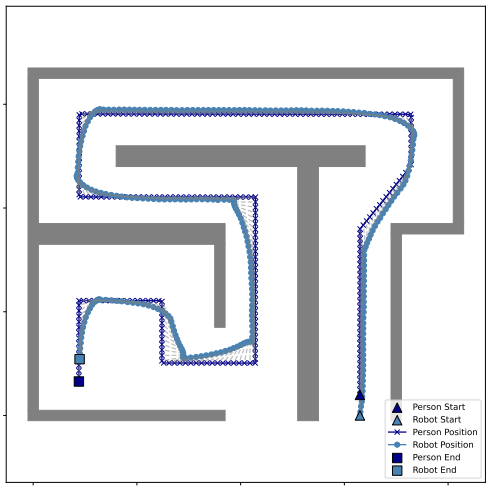
(b) Scenario 1 - Return Trajectory



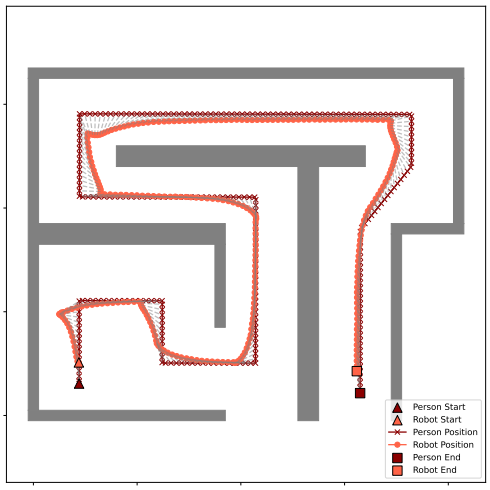
(c) Scenario 2 - Outbound Trajectory



(d) Scenario 2 - Return Trajectory



(e) Scenario 3 - Outbound Trajectory

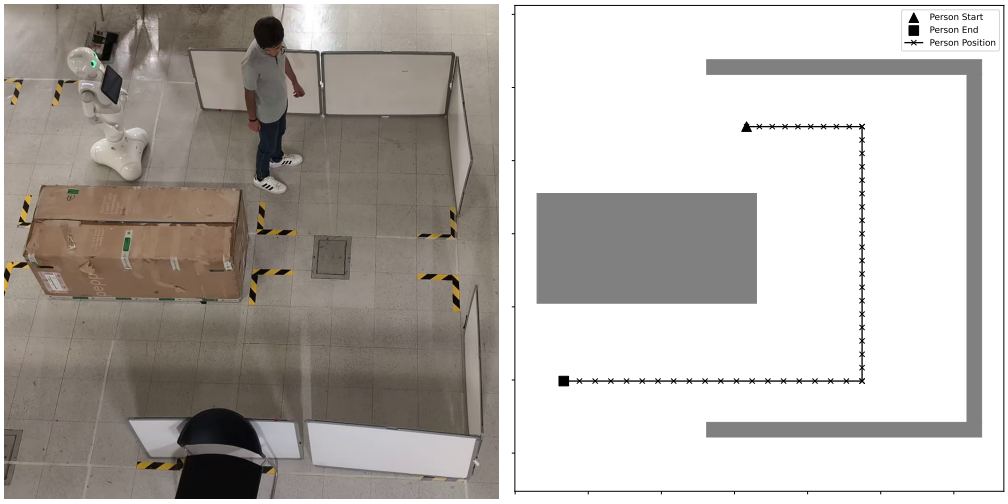


(f) Scenario 3 - Return Trajectory

Fig. 14: Results showing the trajectories of Pepper and the person in three scenarios. Blue lines represent the outbound paths, while red lines indicate the return paths. Grey lines connect corresponding points at the same time instant between the person and the robot's trajectories.

4.3.1 Results of Integration with Physical Robot

After evaluating the optimal control policy obtained through reinforcement learning via simulation, we proceeded to integrate it using Naoqi and ROS to assess the complete functionality with the physical robot. To evaluate the robot's autonomous tracking capability, we designed a scenario similar to Scenario 1 tested in the simulation, wherein the robot was required to follow a person from one end to the other. This environment is depicted in Figure 15.



(a) Physical Scenario **(b)** Proposed Person Trajectory

Fig. 15: Scenario for Physical Testing

In this scenario, Pepper was able to autonomously follow the person from one end to the other while maintaining the predetermined safety distance. The trajectories followed by both the person and the robot are illustrated in Figure 16. Additionally, a video of this experiment is available in the footnote. However, during the tests, a significant issue was observed concerning the tracking efficiency in the real-world integration, unlike in the simulation environment. This issue pertained to the robot's tracking speed; the robot required the person to walk relatively slowly to effectively maintain tracking. The maximum speed that the policy had learned to use was insufficient for keeping pace with the person. This discrepancy can be attributed to several factors, including differences between the simulation and real-world environments, such as differences in simulation time versus real time, and variations in the robot's speed due to factors like friction or surface types.

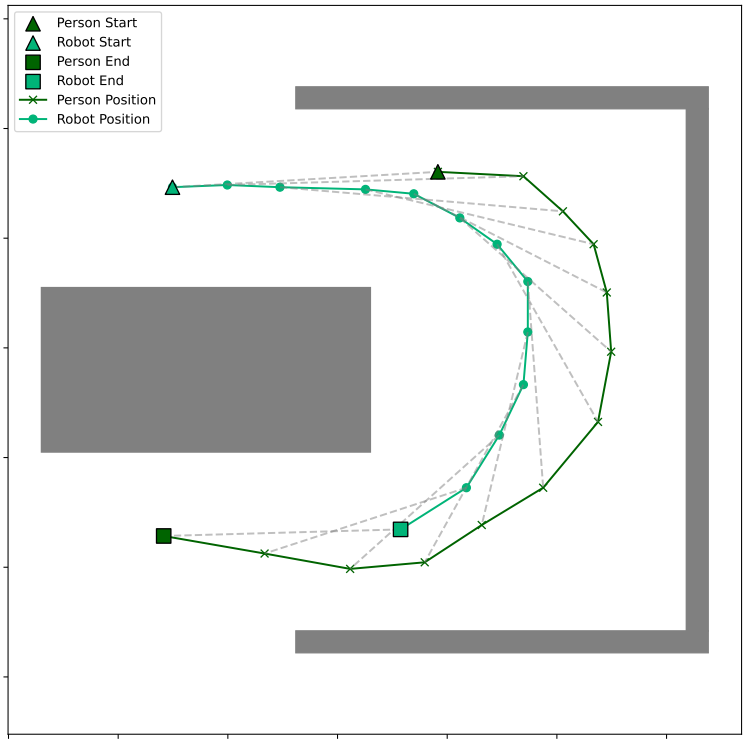


Fig. 16: Results of Person and Robot Trajectories in the Physical Scenario Experiment

Therefore, for future work, it is recommended to fine-tune these parameters during the transition to a real environment or even conduct subsequent training steps with the actual robot. This would enable the policy to autonomously adjust these parameters. Despite the limitations and differences observed between the simulation and real-world results, the experiments demonstrated that the proposed modular architecture effectively endowed Pepper with autonomous person-following capability. With additional effort, these limitations can potentially be overcome, achieving a perfect adaptation.

4.4 Recovery Behavior

In this study, we introduced a recovery behavior module designed as an emergency mechanism for instances when the subject is lost from the tracking performed by the detection and tracking module. Despite the absence of such occurrences in previous experiments, which underscores the robustness of the base architecture and the efficacy of the reinforcement learning-derived policy, the development and implementation of this module are crucial. This preparation ensures system readiness for worst-case scenarios, which are likely in complex real-world environments.

To evaluate the effectiveness of this module, particularly the initial action of the protocol aimed at autonomously resuming tracking, we designed an experiment. The

experiment simulated scenarios where the tracked individual abruptly exited Pepper's field of vision. The robot then executed the recovery maneuver, rotating about its axis to reacquire tracking of the subject using the StrongSORT++ algorithm. The subject was moved rapidly to a random distance between 0.5m and 2m and at a random angle between 60 and 180 degrees relative to the robot, repeated across 100 trials. As depicted in Figure 17, a pie chart illustrates the distribution of successful tracking recoveries.

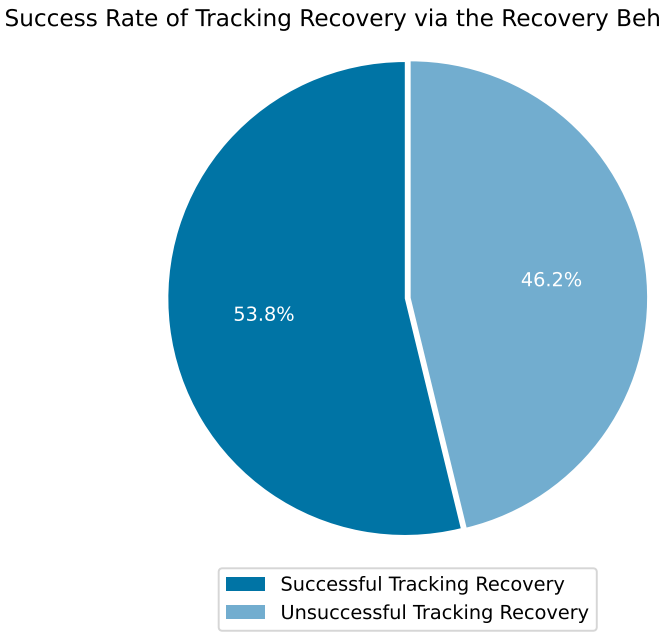


Fig. 17: Success Rate of Tracking Recovery via the Recovery Behavior Module

The results indicate that in 53.8% of cases, the first action of the recovery mechanism successfully reestablished tracking, allowing autonomous following behaviors to continue. It was observed that the recovery capability was directly dependent on the angle at which the subject reappeared. This angle influenced the time taken for the robot to relocate the subject, a critical factor for the StrongSORT++ tracker to recognize the subject as the same previous followed entity.

It is important to note that in instances where this initial action failed (46.2% of cases in this experiment), the protocol's second stage necessitated that the subject approach the robot and physically interact by touching its head. While this method is infallible, it is less desirable as it disrupts the autonomy of the tracking process but remains necessary to ensure continuity in tracking.

5 Conclusion and Future Work

This research advances the autonomous person-following capabilities of social robots through a novel architecture that integrates Convolutional Neural Networks and Reinforcement Learning, as demonstrated on a Pepper-type robot. Our findings highlight the robust performance of our detection and tracking module, which employs YOLO-NAS and StrongSORT++ for high accuracy even under environmental challenges such as occlusions and crowded settings. The 3D Pose Estimation module ensures precise distance measurements, enabling the robot to maintain a safe following distance. Moreover, the Robot Control module, which utilizes Proximal Policy Optimization, allows for adaptive movement control in response to dynamic interactions with the followed person across various challenging environments. The Recovery Behavior module enhances system robustness in real-world applications by enabling the re-establishment of tracking when visual contact is temporarily lost. Collectively, these modules demonstrate the capability and robustness of our proposed modular architecture, empowering a social robot to autonomously follow a person using only a 2D camera and a 3D sensor as input devices. These results significantly advance the interaction of social robots within human-centric environments.

Looking ahead, we propose several enhancements to further refine our system. Firstly, implementing an additional module for evasion of both static and dynamic obstacles is essential. This development will enable the robot to respond more adeptly to unexpected events and adjust its tracking trajectory to navigate around obstacles while preserving its following behavior. Integrating advanced sensors, such as lasers and sonars, into the reinforcement learning algorithm would enrich the robot's environmental perception, facilitating more sophisticated interactions.

Additionally, future research should focus on refining the transition from simulation training to real-world applications. This step is crucial for ensuring that the theoretical benefits observed in controlled environments translate effectively to practical, everyday scenarios. Furthermore, testing our architecture across a broader range of robot models could help generalize these advancements, making the benefits of our research more widely accessible in the field of social robotics.

Moreover, we plan to extend the Recovery Behavior module to include more complex behaviors for reacquiring person tracking. Inspired by the methodologies presented by Algabri and Choi (2020), this enhancement would enable the robot to autonomously move to the last known position of the person if tracking is lost, thereby initiating a systematic recovery protocol. These advancements are poised to not only enhance the robot's adaptability and efficiency in real-world scenarios but also improve its capability to manage complex interactions in dynamic social environments.

References

Shay Aharon, Louis-Dupont, Ofri Masad, Kate Yurkova, Lotem Fridman, Lkdci, Eugene Khvedchenya, Ran Rubin, Natan Bagrov, Borys Tymchenko, Tomer Keren, Alexander Zhilko, and Eran-Deci. Super-gradients, 2021. URL https://zenodo.org/record/7789328.

Redhwan Algabri and Mun-Taek Choi. Deep-learning-based indoor human following of mobile robot using color feature. *Sensors*, 20(9), 2020. ISSN 1424-8220. doi: 10.3390/s20092699. URL https://www.mdpi.com/1424-8220/20/9/2699.

Masashi Awai, Takahito Shimizu, Toru Kaneko, Atsushi Yamashita, and Hajime Asama. Hog-based person following and autonomous returning using generated map by mobile robot equipped with camera and laser range finder. In Sukhan Lee, Hyungsuck Cho, Kwang-Joon Yoon, and Jangmyung Lee, editors, *Intelligent Autonomous Systems 12*, pages 51–60, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-33932-5.

Maxime Busy and Maxime Caniot. qibullet, a bullet-based simulator for the pepper and nao robots. *arXiv preprint arXiv:1909.00779*, 2019.

Bao Xin Chen, Raghavender Sahdev, and John K. Tsotsos. Integrating stereo vision with a cnn tracker for a person-following robot. In Ming Liu, Haoyao Chen, and Markus Vincze, editors, *Computer Vision Systems*, pages 300–313, Cham, 2017. Springer International Publishing.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Peter Dayan and Yael Niv. Reinforcement learning: The good, the bad and the ugly. *Current opinion in neurobiology*, 18:185–96, 09 2008. doi: 10.1016/j.conb.2008.08.003.

Maartje de Graaf, Somaya Allouch, and Jan A.G.M. Van Dijk. What makes robots social?: A user's perspective on characteristics for social human-robot interaction. 10 2015. ISBN 978-3-319-25553-8. doi: 10.1007/978-3-319-25554-5_19.

Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again, 2023.

Beatriz Quintino Ferreira, Kelly Karipidou, Filipe Rosa, Sofia Petisca, Patrícia Alves-Oliveira, and Ana Paiva. A study on trust in a robotic suitcase. In Arvin Agah, John-John Cabibihan, Ayanna M. Howard, Miguel A. Salichs, and Hongsheng He, editors, *Social Robotics*, pages 179–189, Cham, 2016. Springer International Publishing. ISBN 978-3-319-47437-3.

Juan José García Cárdenas. Ópera aprende a jugar linea 4 con aprendizaje por refuerzo. Technical report, Universidad de los Andes, 2019. URL http://hdl.handle.net/1992/44471.

César Daniel Garrido Urbano. Sistema de navegación para robot móvil basado en aprendizaje por refuerzo. Technical report, Universidad de los Andes, 2020. URL http://hdl.handle.net/1992/48862.

Cesar Luis González Gutiérrez. Sistema de navegación autónoma para robot pepper basado en aprendizaje por refuerzo. Technical report, Universidad de los Andes, 2022. URL http://hdl.handle.net/1992/64472.

Frank Hegel, Claudia Muhl, Britta Wrede, Martina Hielscher-Fastabend, and Gerhard Sagerer. Understanding social robots. pages 169–174, 02 2009. doi: 10.1109/ACHI.2009.51.

Noriyuki Kawarazaki, Lucas Tetsuya Kuwae, and Tadashi Yoshidome. Development of human following mobile robot system using laser range scanner. *Procedia Computer Science*, 76:455–460, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.12.310. URL https://www.sciencedirect.com/science/article/pii/S1877050915038119. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015).

Dat Nguyen Ngoc, Valerio Ponzi, Samuele Russo, and Francesco Vincelli. Supporting impaired people with a following robotic assistant by means of end-to-end visual target navigation and reinforcement learning approaches. In *ICYRIME 2021 International Conference of Yearly Reports on Informatics Mathematics, and Engineering 2021*, CEUR workshop proceedings, July 2021.

OpenAI. Openai baselines: Proximal policy optimization. https://openai.com/index/openai-baselines-ppo/, 2018a. Accessed: 2024-06-01.

OpenAI. Spinning up in deep reinforcement learning. https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html, 2018b. Accessed: 2024-06-01.

Juan Esteban Padilla Torres. Detección de emociones utilizando aprendizaje de máquina multimodal para mejorar la interacción humano-robot de pepper. Technical report, Universidad de los Andes, 2023. URL http://hdl.handle.net/1992/68683.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

Juan Andrés Romero Colmenares and Luccas Rojas Becerra. Improving autonomy and natural interaction with a pepper robot through the evaluation of different large language models. Technical report, Universidad de los Andes, 2023. URL https://hdl.handle.net/1992/73186.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Avinash Kumar Singh, Neha Baranwal, and Kai-Florian Richter. An empirical review of calibration techniques for the pepper humanoid robot's rgb and depth camera. In Yaxin Bi, Rahul Bhatia, and Supriya Kapoor, editors, *Intelligent Systems and Applications*, pages 1026–1038, Cham, 2020. Springer International Publishing. ISBN 978-3-030-29513-4.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Hamid Taheri and Seyed Rasoul Hosseini. Deep reinforcement learning with enhanced ppo for safe mobile robot navigation, 2024.

Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4): 1680–1716, November 2023. ISSN 2504-4990. doi: 10.3390/make5040083. URL http://dx.doi.org/10.3390/make5040083.

Amari Tomoya, Satoru Nakayama, Atsushi Hoshina, and Midori Sugaya. A mobile robot for following, watching and detecting falls for elderly care. *Procedia Computer Science*, 112:1994–2003, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.08.125. URL https://www.sciencedirect.com/science/article/pii/S1877050917314825. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.

Ultralytics. Yolo nas: Modelos y arquitecturas. Ultralytics Documentation, 2023. URL https://docs.ultralytics.com/es/models/yolo-nas/.

Libing Yang, Yang Li, and Long Chen. Clothppo: A proximal policy optimization enhancing framework for robotic cloth manipulation with observation-aligned action spaces, 2024.