Assignment 1 SoC

Fatima Jahara
Department of Computer Science
Rutgers University
fatima.jahara@rutgers.edu

Adel Khorramrouz
Department of Computer Science
Rutgers University
a.khorramrouz@rutgers.edu

Robert Contofalsky
Department of Psychology
Rutgers University
robert.contofalsky@rutgers.edu

1 Task 1: WALKER2D ENVIRONMENT

To build and train our Walker2D agent, we first set up the environment in Google Colab. We installed MuJoCo, Gymnasium, and Stable-Baselines3. MuJoCo is the physics simulation engine required for Walker2d-v4, Gymnasium is the OpenAI Gym environment wrapper and Stable-Baselines3 is a library that provides preimplemented RL algorithms, including SAC.

The walker in the Walker2D-v4 environment is a two dimensional two-legged agent figure consisting of four main body parts: a single torso at the top with two legs splitting after the torso, two thighs in the middle below the torso, two legs in the bottom below the thighs, and two feet attached to the legs on which the entire body rests. The environment is build on top of the hopper environment Durrant-Whyte et al. (2012) allowing the robot to work forward using a new set of legs in MuJoCo (Gymnasium). The goal of the walker is to move in the forward (right) direction by coordinating both sets of feet, legs, and thighs and by applying torques on the six hinges connecting the six body parts.

1.1 Observation Space

The observation space consists of 17 continuous values representing the state of the Walker2D robot that are divided into 2 parts:

1. **qpos**: Consists of 8 elements representing the position values of the robot's body parts.
2. **qvel**: Consists of 9 elements representing the velocities of these individual body parts.

By default, the observation space is Box(-inf, inf, (17,), float64) consisting of 17 continuous state variables that provide information about the agent's physical state at every timestep. At each timestep, the Soft Actor-Critic (SAC) model receives the 17-dimensional observation vector and uses it to decide the optimal torques to apply to the joints which allows the Walker2D agent to learn an effective locomotion strategy.

1.2 Action Space

The action space for Walker2D is box type with continuous action space. The action space is defined by Box(-1, 1, (6,), float32) representing 6 floating point continuous values between -1 and 1 that control the joint torques and walk. A positive value represents the action of moving forward, a negative value represents the action of moving backward, and a zero value means no movement. The agent learns how to coordinate the torques in all six joints to balance and move forward without falling.

At every timestep, the agent selects a six-dimensional action vector, where each value corresponds to the torque applied to a specific joint, allowing precise control over movement. A well-optimized policy ensures that torque values are applied efficiently for stable movement. However, poor action can lead to inefficient movement, unnecessary energy consumption, and loss of balance, causing the agent to fall.

Table 1: Walker2D Action Space

Index	Controlled Joint	Action
0	thigh_joint (Right)	Torque applied on the thigh rotor to moves the right thigh forward/backward.
1	leg_joint (Right)	Torque applied on the leg rotor to Move the right lower leg.
2	foot_joint (Right)	Torque applied on the foot rotor to move the right foot.
3	thigh_left_joint	Torque applied on the left thigh rotor to move the left thigh forward/backward.
4	leg_left_joint	Torque applied on the left leg rotor to move the left lower leg.
5	foot_left_joint	Torque applied on the left foot rotor to move the left foot.

1.3 Rewards

The reward function in the Walker2D environment encourages forward movement while penalizing falls and excessive energy consumption. The reward consists of two rewards and one cost:

1. health_reward: The fixed reward the agent receives for being alive at every timestep.
2. forward_reward: The reward the agent receives for walking forward. This reward is positive if the walker walks forward (right).
3. control_cost: A negative reward or cost to penalize the walker if it takes actions that are too large causing excessive energy consumption.

Total rewards = healthy_reward + forward_reward - control_cost

At each timestep, the agent receives a health reward, which is a fixed positive reward simply for moving forward efficiently and continuing the episode. This component encourages the agent to maintain balance and avoid falling, as episodes terminate prematurely if the agent collapses. In addition to staying alive, the agent is rewarded based on its forward velocity through the forward reward. This reward is directly proportional to the agent's forward movement in the rightward (positive x-axis) direction. If the agent moves forward steadily, it accumulates higher rewards, while slow or erratic movement results in lower rewards. Additionally, a control cost penalty is applied to penalize for high energy consumption to ensure that the agent learns an energy-efficient way of handling torques and avoids applying excessively large torques to the joints. This cost is typically proportional to the sum of squared torques applied to the joints, discouraging abrupt and unnecessary movements. This encourages the development of an energy-efficient walking style, where the agent learns to use minimal effort while maintaining forward momentum. By balancing positive rewards for forward motion with penalties for instability and inefficient energy use, the reward structure guides the SAC agent toward learning an optimal walking strategy that is both fast and energy-efficient while minimizing the risk of falling.

2 Task 2: Soft Actor Critic SAC RL AgenT

To train our Walker2D agent we used the Soft Actor-Critic (SAC) reinforcement learning algorithm to optimize its policy through interaction with the environment. Soft actor-critic (SAC) is a model-free off-policy maximum entropy actor-critic algorithm that avoids the complexity and potential instability associated with approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning Haarnoja et al. (2018).

2.1 Model Training

We first started with our baseline model, where we used the multi-layer perceptron (MLP) policy network which acts both as an actor and a critic network. Given that the input to our SAC model in Walker2D is a vector of numbers, i.e., the observation space that consists of joint angles, velocities, and positions, but not images we used the MLP network instead of Convolutional Neural Network. First, we trained our baseline model, where we used the multi-layer perceptron (MLP) policy network which acts both as an actor and a critic network. The MLP network consists of one input layer, two hidden layers, and one output layer. The input layer takes the state representation from the environment, the hidden layers consists of 256 neurons each and uses the ReLU activation for introducing non-linearity. The output layer of the actor network outputs mean and standard deviation for each action dimension and the critic network outputs Q-values estimating future rewards.

2.1.1 Training Configuration

We trained our agent for 500,000 steps and logged key training metrics, including reward trends, actor and critic losses, and entropy coefficients, using TensorBoard to monitor learning progress.

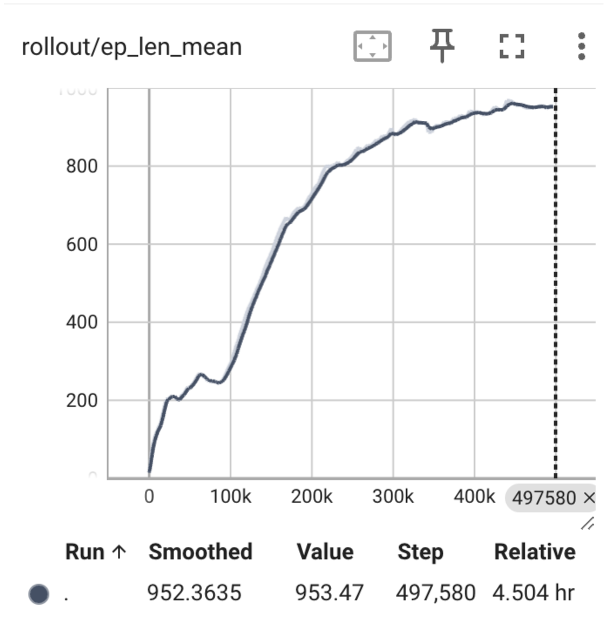
We used the default hyperparameters used by the SAC model for training the baseline model as shown in Figure 2.

Table 2: Baseline SAC agent Hyperparameters

Hyperparameter	**Value**	**Description**
Optimizer	Adam	Optimizes to the most effective policy
Learning Rate (α)	3×10^{-4}	Step size for gradient updates
Replay Buffer Size	10^6	Number of stored transitions for replay
Discount Factor (γ)	0.99	Future reward discounting
Hidden Layers	$[256, 256]$	Two-layer perceptron for policy and critic networks
Target Smoothing Coefficient	0.005	Balances the stability of target Q-network
Entropy Coefficient	$auto$	Balances between exploration and exploitation
Batch Size	256	Balances between stability and speed

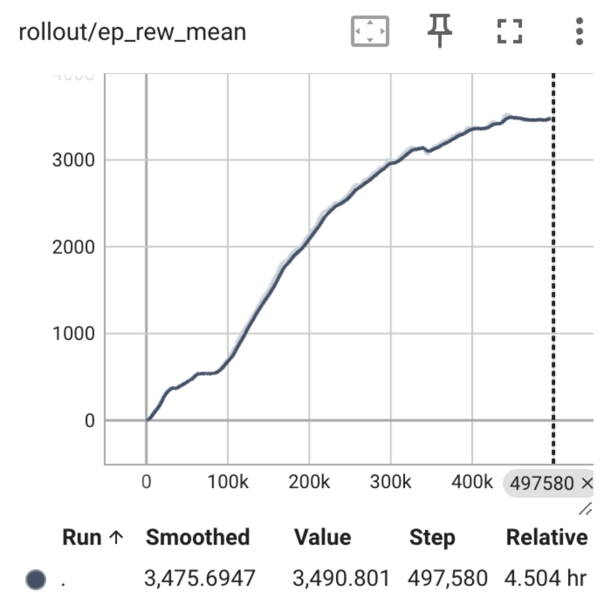
2.1.2 Training Performance and Convergence

Figure 1 shows the average number of steps per episode and the average total reward per episode during training. Here, we can notice that the average episode length, i.e., the average number of steps per episode is 953.47 with no frequent pitfalls and steady step size after 400k episodes gaining near optimal performance. The average reward per episode shows a similar pattern with an average reward of 3,475.69.



Run ↑	Smoothed	Value	Step	Relative
● .	952.3635	953.47	497,580	4.504 hr

(a) Average number of steps per episode



Run ↑	Smoothed	Value	Step	Relative
● .	3,475.6947	3,490.801	497,580	4.504 hr

(b) average total reward per episode

Figure 1: Training Baseline Model

The loss graph in Figure 2 shows how the actor loss aka policy network loss, the critic loss aka Q-value loss, and the entropy coefficient loss changes overtime during training over the 500000 timesteps. Here, the actor loss steadily decreases overtime suggesting good policy optimization, the critic loss oscillates around zero with minor spikes as the critic is learning the Q-values and adjusting over time and the entropy coefficient loss shows almost no major change suggesting that the model maintains a good balance of exploration and exploitation and that the critic is learning correctly.

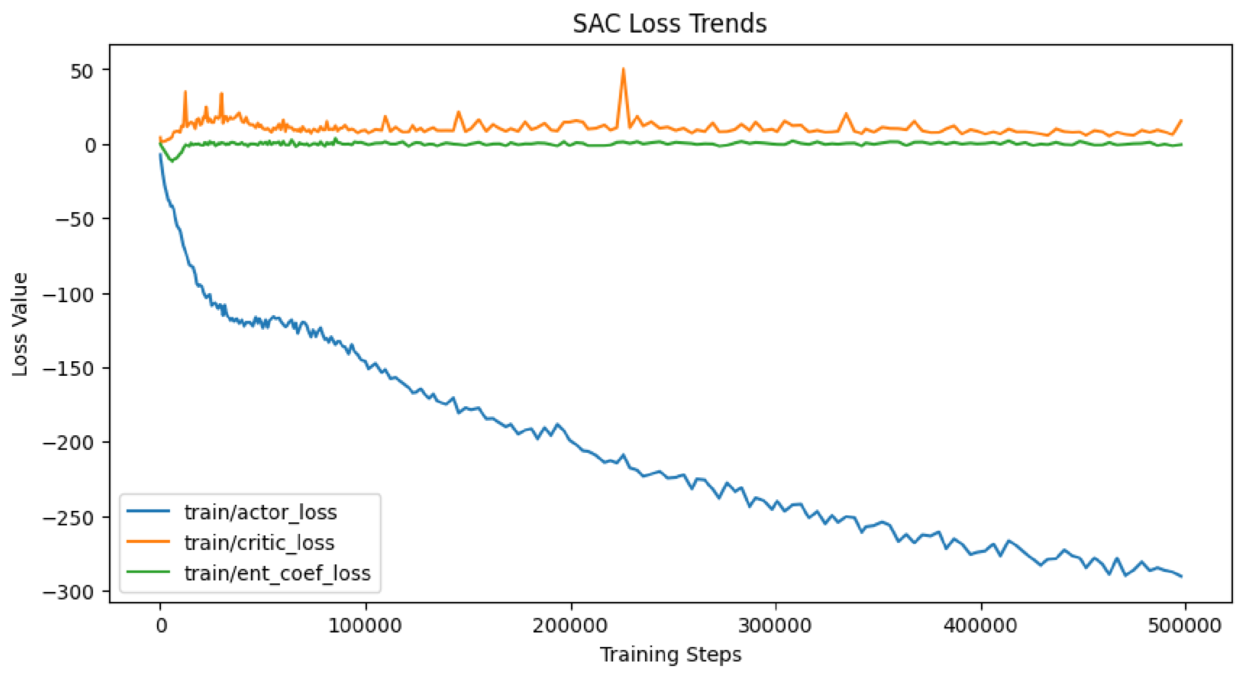
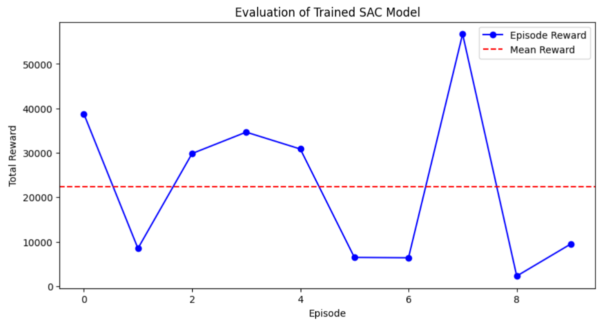


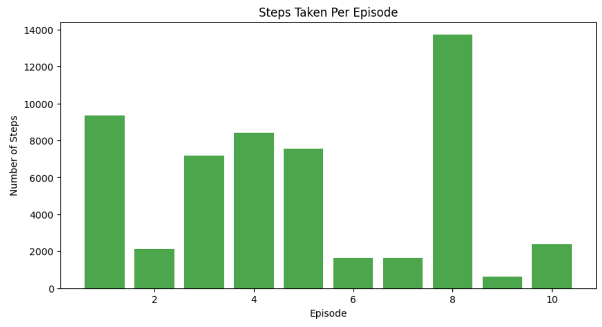
Figure 2: Baseline Model Training Loss

2.2 Baseline Model Evaluation

We evaluated the agent on 10 test episodes and Figure 3 shows how the trained agent performs in each episode. We can see that the agent's performance varies a lot across different test episodes. While some episodes achieve high rewards with a maximum reward of 56718.82, others drop drastically below 10,000, with a minimum reward of 2296.87. The mean reward is 22410.30. Similarly, Figure 3b shows the variance in the number of steps taken per episode with a maximum of 13729 in episode 8 and a minimum of 643 in episode 9. The agent performs well in some cases but fails in others showing lack of consistency. This inconsistency suggests that the model may not generalize well to different test conditions, possibly due to overfitting to specific training scenarios. The high rewards in multiple episodes confirm that the agent has learned effective movement strategies, but the drastic drops in other episodes highlight the need for improved robustness and adaptability.



(a) Reward per Episode



(b) Steps per Episode

Figure 3: Training Baseline Model

The sequence of 20 spase frames out of the early 100 frames is shown in Figure **??** showcases how the trained baseline SAC agent in the Walker2D environment executes movement. Here, one leg contributes more than other. Also, the knee and thigh joints does not seem to work well, especially the thigh rotors. This could be due to an imbalanced reward function, where health reward and forward reward is over-prioritized and thus it tries hopping instead of purely walking by performing this reward hacking. The last 20 frames of the episode depict the complete fall of the Walker2D agent shown in Figure 5, highlighting the failure in stability and tipping over. The agent begins in a forward-leaning position with its legs attempting to touch the ground. However, as seen in these frames, the legs fail to make contact with the ground in time, leading to both feet losing ground contact. Instead of attempting to stabilize by extending a leg forward, the agent remains in a rigid posture, suggesting that the policy lacks a corrective mechanism for balance. The instability and the reward hacking can be due to several reasons, including the reward function being biased towards forward reward and health reward, insufficient control cost penalties, policy not being trained properly for foot placement causing decreased use of some joints, uncontrolled forward fall, and termination.

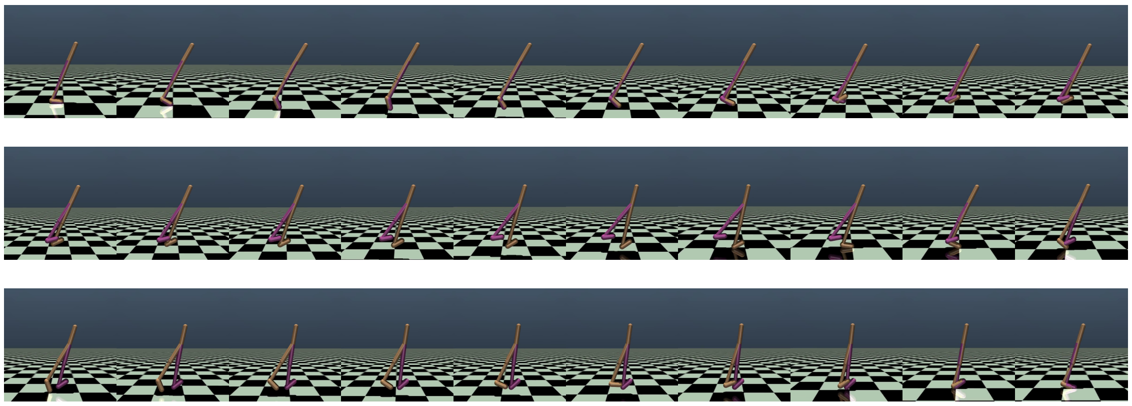


Figure 4: Visualization of the agent's movement from frames 30 to 59, capturing learned policy.

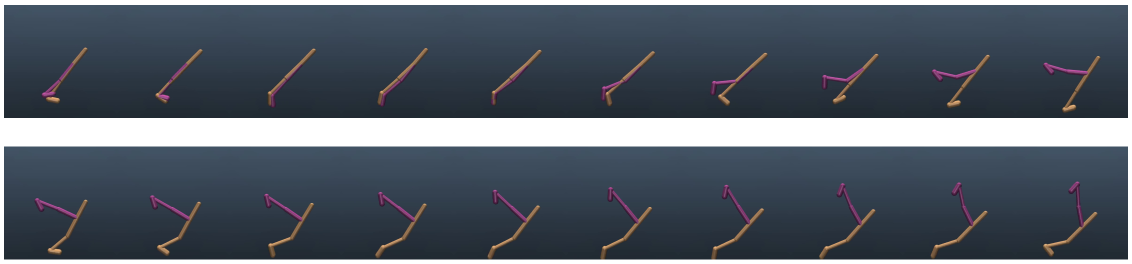


Figure 5: Visualization of the agent's learned policy failure (20 extracted frames as it falls).

2.3 Hyperparameter Tuning and Optimization

The inconsistencies in the agent's performance make it clear that some key hyperparameters need fine-tuning to improve stability and generalization to different observation settings. While the agent has learned to move forward effectively in certain episodes, its tendency to hop, lose balance, and fall suggests that it might be exploiting the reward function rather than learning robust walking behavior. To address this we tried tuning different sets of hyperparameters. We evaluated the models on the hyperparameter space with a maximum of 70000 steps and 2 episodes given some of the episodes takes longer that 5000,000 steps which is time and resource consuming. Also, we evaluated the agents using the same observation space for better evaluation. Figure 6 shows evaluation of the policies using different hyperparameters.

The hyperparameters we tuned, their search space, and optimal parameters, are given below:

1. **Number of hidden layers and Number of hidden units per layer:** To see the impact of model complexity we tried varying the number of model layers and number of neuron per layers to see how increasing complexity effect the learned policy and how the agent perform. The different layer and hidden unit configurations we tries are: [256, 256] (baseline), [512, 256], [512, 256, 128], [256, 256 218], and [128, 128, 64]. From our experiment we found that, [512, 256] and [256, 256] perfromed almost similar with less stability. But [512, 256, 128] and [256, 256 218] were more stable.
2. **Learning Rate:** A high learning rate can make training unstable, causing the agent to take large, erratic updates that prevent it from converging to a stable walking behavior. On the other hand, a low learning rate can slow down learning significantly, making it difficult for the agent to refine its movement strategy within a

reasonable number of training steps. The default learning rate used by our baseline model was 0.0003. The different learning rates we tuned to are: 0.00003 and 0.000003. Here, we found that agents performed better as their learning rate increased.

3. **Target Smoothing Coefficient:** Target smoothing coefficient controls the soft target network update rate helps in stable learning. A higher tau allows the target network to adapt more quickly, but this can introduce instability by making the updates too reactive to recent experiences. A lower tau results in slower updates, which stabilizes training but may slow down convergence, requiring more training steps for the agent to refine its walking strategy. Our default model used a coefficient value of 0.005. We tuned our model on coefficient value of 0.001 to see if that help in stabilizing the strategy and we found that agents performed better with 0.001 tau.

4. **Discount Factor:** Tuning the discount factor helps balancing short-term and long-term rewards in SAC. A high gamma makes the agent prioritize long-term rewards, which can encourage forward movement but may lead to instability if the agent sacrifices balance to maximize cumulative reward. On the other hand, a lower gamma places more emphasis on immediate rewards, which can help the agent focus on short-term stability and prevent reckless forward-leaning behavior that leads to falls. Our baseline model uses 0.99 and we tuned our model for a value of 0.999 and 0.95 and found that, higher the discount factor the better.

5. **Energy Coefficient:** Tuning the entropy coefficient in SAC helps balance between exploration and exploitation. When set to "auto", the agent dynamically adjusts its exploration level. We tuned our agent to a value of 0.1 to reduce exploration for more deterministic actions and stabilize walking patterns and it did help the agent perform better.

6. **Batch Size:** A larger batch size allows the agent to make more stable and generalized updates by averaging over a broader set of experiences. A smaller batch size leads to faster updates but may introduce more variance in learning, potentially causing unstable behaviors like hopping or inconsistent gait patterns. Thus, we tuned our agent on a larger batch size of 512 only apart from the default batch size 256 and found that the model performed better with a batch size of 512.

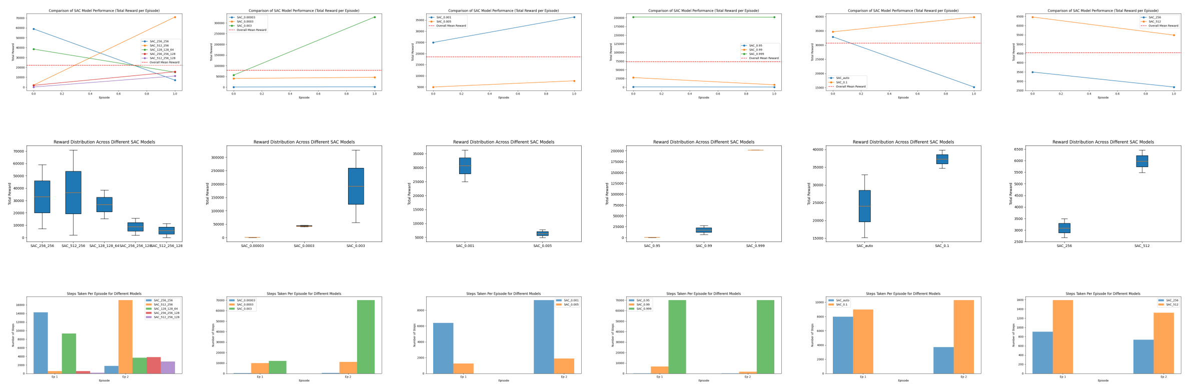


Figure 6: Sensitivity of soft actor-critic agents to selected hyperparameters on Walker2D.

Hence, for our final model we plan to use the [512, 256, 128] architecture, as it showed better stability compared to simpler models while maintaining strong performance with a learning rate of 0.0003, target smoothing coefficient of 0.001, discount factor of 0.999, entropy coefficient of 0.1, and batch size of 512, since these hyperparameters improved stability, long-term reward prioritization, and overall agent performance. Given the best working parameters and the best models with each parameter saved we plan to develop our final agent by trying the following two techniques:

1. Training a new model with the optimal hyperparameters.

2. Using knowledge distillation to train a new SAC agent (student) using the best performing pre-trained SAC models (teachers).

3 Task 3: FEASIBLE APPLICATION

3.1 Designing a Socially Cognizant RL Agent with Theory of Mind for Autonomous Trash Collection

Urban environments often face challenges in managing waste efficiently, leading to environmental pollution and health hazards. Furthermore, improper trash collection can result in water and soil contamination, posing significant risks to public health United Nations Environment Programme (2015). Autonomous trash collection robots have been proposed as a solution to mitigate these issues by identifying and collecting waste in public spaces. For instance, researchers have developed robots equipped with cameras to detect waste and navigate urban areas effectively Ganesan et al. (2017). These robots have been deployed in various settings, including university campuses and public plazas, to study human-robot interactions and encourage proper waste disposal Cornell Tech (2019).

However, to develop socially cognizant trash collection robots, they must navigate complex areas and interact seamlessly with humans. This requires the integration of advanced perception systems capable of understanding human intentions and behaviors. Incorporating a Theory of Mind (ToM) module enables robots to interpret human actions and anticipate needs, facilitating smoother interactions and more efficient waste collection Alfeo et al. (2019).

Given the necessity for socially aware robots in effective waste management, implementing a ToM module allows these robots to comprehend human intentions based on actions and environmental context. This comprehension enhances their ability to operate autonomously in dynamic public spaces, ultimately contributing to cleaner and healthier urban environments.

The proposed robot needs many computer vision components to achieve efficient trash collection and intercepting human ToM, including but not limited to:

1. **Object Recognition:** These robots need to be capable of detecting waste and recycle bins, Trash objects (e.g. drink cans), and people. Various methods using deep learning had been implemented for more accurate classifications Wang et al. (2020).
2. **Semantic Segmentation**: After detecting each of mentioned classes it is necessary that they can differentiate each component (e.g. trash) from the surrounding environment, allowing the robot to focus on relevant objects and take actions accordingly Bashkirova et al. (2021).
3. **Object Detection**: Detects specific trash objects like bottles, cans, and papers, enabling targeted collection efforts Bashkirova et al. (2021).
4. **Pose Estimation**: To enable the ToM module, an important step is to recognize human postures to determine if a person is holding trash or reaching for a bin, enhancing interactive capabilities Li et al. (2023). Further, that is used for understanding human navigation and human-robot collision avoidance Zhang et al. (2023).
5. **Action Recognition**: To identify human behaviors such as littering or handing trash to the robot, allowing appropriate responsive actions Wang et al. (2019).

3.1.1 System Architecture

1. **Perception Module** (Input: Real-world scene captured by the camera. Output: Raw image frames.)

   This module collects visual data from cameras to pass to the second unit.
2. **Pre-processing module:** (Input: Raw frames from Perception Module. Operation: Convert input image to specified format/size. Output: Reformatted frames.)

   In this module, input is the frames from the camera and we convert input image from camera to a specified format and size requested for each subsection in the computer vision processing unit. For instance, we transform RGB format images into HSI color format in order to reduce the effect of environmental brightness for the trash bin detection task Yu (2020).
3. **Computer Vision Processing Unit**: (Input: Pre-processed frames. Output: (a) Positions of trash bins, (b) Detection of trash, (c) Material classification, (d) Action recognition signals.)
   (a) **Trash bin detection:** Following Yu (2020), we transform RGB captured images to HSI color space to mitigate factors such as environmental brightness. By analyzing the saturation and hue values, specific thresholds are established for image segmentation. This process further enables the accurate determination of trash bin positions.
   (b) **Trash (garbage) Detection:** A significant step for our robot is accurately distinguishing garbage from non-garbage objects. Wang et al. (2021) provides a deep learning model capable of real-time garbage

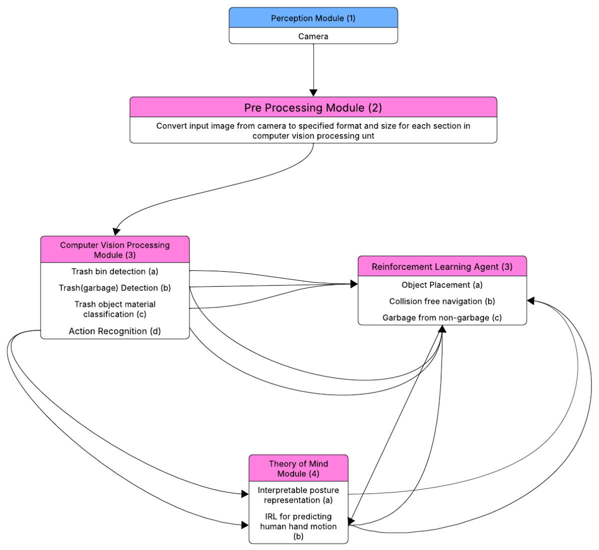

Perception Module (1)
Camera
Pre Processing Module (2)
Convert input image from camera to specified format and size for each section in computer vision processing unt
Computer Vision Processing Module (3)
Trash bin detection (a)
Trash(garbage) Detection (b)
Trash object material classification (c)
Action Recognition (d)
Reinforcement Learning Agent (3)
Object Placement (a)
Collision free navigation (b)
Garbage from non-garbage (c)
Theory of Mind Module (4)
Interpretable posture representation (a)
IRL for predicting human hand motion (b)

Figure 7: System architecture for socially cognizant trash collection robot.

detection. To evaluate the effectiveness of proposed approach, we compare the detection results against state-of-the-art models, as summarized in Table 3. From the detection results, it can be observed that under different conditions, Faster R-CNN, YOLOv3, and the proposed algorithm can detect various types of garbage objects. Tthe overall confidence level of Wang et al. (2021) algorithm is higher compared to Faster R-CNN and YOLOv3. Also it demonstrates a reduced computational time compared to Faster R-CNN and YOLOv3, with a single image processing time of 25ms.

Table 3: Mean Average Precision (mAP) and Time Results Comparison Between Different Models Which is essential for real-time desicion making

Model	Detection Time (s)	mAP (%)
Faster R-CNN	0.116	86.86
YOLOv3	0.032	89.69
Ours	0.025	95.36

(c) **Trash object material classification:** Kunwar (2023) introduce MWaste, a mobile application that leverages computer vision to classify household waste into categories such as trash, plastic, paper, metal, glass, or cardboard. The application utilizes neural network architectures to process real-world images of waste items, achieving an average precision of 92% on the test set. We will use the MWaste approach to classify the type of trash detected in the first place.

(d) **Action Recognition:** To enable the ToM module, we need to recognize human postures and use it to predict actions accordingly as done in Freedman et al. (2022). The authors propose a symbolic representation derived from visual sensor data. Further, Arathorn (2015) presents a computational model that integrates visual and motor areas to recognize and interpret human shapes, poses, and actions. The model unifies solutions to inverse problems such as visual transformation discovery and inverse kinematics, facilitating the understanding of human behavior.

4. **Theory of Mind Module**: (Input: Outputs from the Computer Vision Processing Unit. Output: Inferred human intentions and predicted actions.)

   Based on the processed data in the prior unit (Computer Vision Processing), this unit may infer human intentions and predict their actions.

   (a) **Interpretable posture representation:** introduced by Freedman et al. (2022) can serve as a foundation model for developing Theory of Mind capabilities in AI systems, enabling them to infer human intentions and mental states based on observed postures.

   (b) **IRL for predicting human hand motion:** Based on Jara-Ettinger (2019), AI systems infer the goals and intentions of agents by observing their actions, treating behavior as a rational attempt to maximize utility.

On the other hand, Inverse Reinforcement Learning (IRL) is used for predicting human hand motion and intended targets during human-robot handover tasks as proposed by Mainprice and Berenson (2013). In our application, understanding the hand motion has dual benefits. Initially, it can help to learn garbage detection from non-garbage and define rewards for the garbage detection module. Secondly, when it detects garbage in human hands, it can infer the human goal is to put it in the trash bin. (Input is human actions from CV and posture interpretation from ToM, output is a predicted intention.)

5. **Reinforcement Learning Agent**: (Input: Computer Vision outputs + ToM inferences. Output: Robot's control actions for navigation, object placement, etc.)

   Makes decisions based on Computer Vision and ToM outputs to navigate the environment and interact with humans, trash bins, and trash objects effectively. Two different RL networks need to be trained, briefly discussed here:

   (a) **Object Placement:** In completion of Wang et al. (2021) that uses pure computer vision for object detection, we will use Herzog et al. (2023) method. Their method gives positive feedback (reward) when the model accurately and successfully identifies and places an item into the correct waste bin.

   (b) **Collision free navigation:** In order to keep our human-robot coexistence in public places such as campus safe, we follow Sangiovanni et al. (2018) approach. This study presents a real-time collision avoidance method for robotic manipulators operating in environments shared with humans. The approach employs Deep Reinforcement Learning (DRL) techniques to enable robots to learn collision-free trajectories, enhancing safety in human-robot coexistence scenarios.

   (c) **Garbage Detection:** Our RL module learns the garbage detection task better utilizing the image classification method proposed in Wang et al. (2021). That is further used as both input and output of the ToM module as shown in Figure 7 and explained in subsubsection 4b. (Here, the RL agent refines its policy using inferred human intentions for improved garbage-vs-non-garbage identification.)

The integration of advanced computer vision techniques and RL algorithms has been demonstrated in various studies. The specific sub-task of trash collection has been studied and for waste identification has shown promising results in enhancing waste management systems Wang et al. (2020). Additionally, the development of datasets like ZeroWaste facilitates improved segmentation and detection of waste objects in cluttered scenes Bashkirova et al. (2021). These advancements support the feasibility of implementing a socially cognizant autonomous trash collection robot equipped with a Theory of Mind module.

4 Task 4: SC DESIGN

4.1 Does the Design improve quality of life (QOL)?

The campus robot trash collector (CRTC) enhances QOL of all who use campus facilities (students, faculty, employee & visitors) by improving sanitation and waste management efficiency. By virtue of this robot actively reducing litter in public places, the robot ensures a cleaner and healthier environment – augmenting public hygiene. Its ability to continuously collect small waste items (plastic bottles, paper, cups, etc) reduces visual pollution and also lessens the burden on janitorial employees that would otherwise spend more time than necessary to pick up trivial waste. Additionally, it being asynchronous enables it to be operation in all hours of the day, allowing human workers to not worry about overnight activity in the area. This low-level automation gives more time to janitorial staff to focus on more complex cleaning tasks such as mopping, scrubbing, handling hazardous materials, etc. Doing so improves the overall efficiency of campus waste management and thereby the QOL campus cleaners by eliminating monotonous and time consuming work of litter removal.

4.2 What are the potential unintended consequences?

There are several unintended consequences of CRTCs. The first one would be technical: inefficient sorting. In this case, the robot would have difficulty distinguishing between recycling, general waste, or the natural environment (e.g., misunderstanding a tree-trunk for cardboard). This type of unintended consequence would introduce a redundancy for the programmers, the janitors that might sort out these miscategorizations and even at the waste management companies that handle recycling or waste misallocating the robots decisions. The second unintended consequence would be social: the robots decrease pedestrian safety. While trivial at first glance, these robots could very easily lack the social navigation necessary to operate within a social setting such as a campus, and therefore obstruct pathways, create unexpected hazards (such as running into people). This would in-turn create an environment that decrease trust in robotics and dis-incentivizes their deployment, on top of harming human agency. A third one would be that these robots create an overreliance on automation. Knowing that a robot would collect litter at any point in time could incentivize humans to leave their trash behind because a robot would pick it up for them, and not have a burden of guilt for having this attitude. Ultimately, this would result in a messier campus because this type of unintended consequence would allow humans to accumulate more litter than before. A fourth unintended consequence is the high potential for e-waste. Given that there would need to be a lot of robots deployed on campus to efficiently reduce litter, this would result in many of them potentially breaking down or having them be replaced due to new version or current inefficiencies.

4.3 Are there safety leaders?

The CRTC will have three layers of safety leaders: Technical oversight, Regulatory Oversight, Community Engagement. The first layer, technical oversight, is necessary to monitor the robot's system as a whole (i.e., its sensors, navigation systems, and trash collection efficiency), ensuring that it operates reliably in diverse campus conditions. Engineers and technicians should regularly update its software and troubleshoot any mechanical failures to prevent malfunctions. Regulatory oversight, includes campus authorities and municipal agencies setting operational guidelines to ensure that there are NO unnecessary disputes about cleaning zones, pedestrian interaction protocols, etc. Thirdly, community engagement is crucial to addressing concerns from students and faculty who have daily robot interactions. Incorporating feedback from campus users and maintenance staff, safety leaders helps informed decisions making about robot performance, and ensures it integrates smoothly into campus operations while prioritizing public safety.

4.4 Is there a mechanism for users to provide feedback and is there a plance to act on that feedback?

To ensure that humans are in the loop with the CRTCs, we have designed an app that would allow for students, faculty, and university employees to give feedback on robot performance. This app gives the users an in-depth rating scale of 1-5* and also enables more motivated users to write down more in-depth improvements for the engineers to consider, and not solely rely on the basic 1-5* scale that most robotics app rely on (e.g., Waymo).

4.5 What are the ethical considerations?

The deployment of trash collection robots raises ethical considerations that must be carefully managed: Equitable deployment, job displacement, and environmental impact. In the first case, equitable deployment is a key concern, as these robots should be distributed fairly across all campus areas rather than being concentrated in specific departments (i.e., within ECE/Robotics). Ensuring that all students benefit from cleaner environments is essential for ethical implementation. Second, Job displacement is another issue, as automation in sanitation services could reduce the need for human janitors. However, rather than replacing jobs outright, the robots should be used to complement existing cleaning efforts, allowing human workers to focus on more complex sanitation tasks that require manual intervention. Additionally, **environmental impact** must be considered. While the robot improves waste management in the short term, its long-term sustainability depends on responsible manufacturing, energy-efficient operation, and an effective plan for recycling or repurposing old units to prevent e-waste accumulation.

References

Alfeo, A. L., Castelló Ferrer, E., Lizarribar Carrillo, Y., Grignard, A., Pastor, L. A., Sleeper, D. T., Cimino, M. G. C. A., Lepri, B., Vaglini, G., Larson, K., Dorigo, M., and Pentland, A. (2019). Urban swarms: A new approach for autonomous waste management. In 2019 International Conference on Robotics and Automation (ICRA), pages 4233–4240. IEEE.

Arathorn, D. W. (2015). A system view of the recognition and interpretation of observed human shape, pose, and action. arXiv preprint arXiv:1503.08223.

Bashkirova, D., Abdelfattah, M., Zhu, Z., Akl, J., Alladkani, F., Hu, P., Ablavsky, V., Calli, B., Bargal, S. A., and Saenko, K. (2021). Zerowaste dataset: Towards deformable object segmentation in cluttered scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 0–1.

Cornell Tech (2019). (almost) everyone likes a helpful trash robot.

Durrant-Whyte, H., Roy, N., and Abbeel, P. (2012). Infinite-Horizon Model Predictive Control for Periodic Tasks with Contacts, pages 73–80.

Freedman, R. G., Mueller, J. B., Ladwig, J., Johnston, S., McDonald, D., Wauck, H., Wheelock, R., and Borck, H. (2022). A symbolic representation of human posture for interpretable learning and reasoning. arXiv preprint arXiv:2210.08998.

Ganesan, S., Durgalakshmi, B., and Seyatha, K. (2017). Autonomous trash collecting robot. International Journal of Engineering Research and Technology, 6(4):231–234.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

Herzog, A., Rao, K., Hausman, K., Lu, Y., Wohlhart, P., Yan, M., Lin, J., Gonzalez Arenas, M., Xiao, T., Kappler, D., Ho, D., Rettinghouse, J., Chebotar, Y., Lee, K.-H., Gopalakrishnan, K., Julian, R., Li, A., Fu, C. K., Wei, B., Ramesh, S., Holden, K., Kleiven, K., Rendleman, D., Kirmani, S., Bingham, J., Weisz, J., Xu, Y., Lu, W., Bennice, M., Fong, C., Do, D., Lam, J., Bai, Y., Holson, B., Quinlan, M., Brown, N., Kalakrishnan, M., Ibarz, J., Pastor, P., and Levine, S. (2023). Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators. arXiv preprint arXiv:2305.03270.

Jara-Ettinger, J. (2019). Theory of mind as inverse reinforcement learning. Cognitive Science, 43(6):e12751.

Kunwar, S. (2023). MWaste: A Deep Learning Approach to Manage Household Waste. arXiv preprint arXiv:2304.14498.

Li, Y., Li, W., Zhang, P., Wang, Y., Wang, L., and Liu, L. (2023). Single-view multi-human pose estimation by attentive cross-level fusion with transformers. Frontiers in Neuroscience, 17:1201088.

Mainprice, J. and Berenson, D. (2013). Enhanced human-robot collaboration with intent prediction using inverse reinforcement learning. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 299–306.

Sangiovanni, B., Rendiniello, A., Incremona, G. P., Ferrara, A., and Piastra, M. (2018). Deep reinforcement learning for collision avoidance of robotic manipulators. In 2018 IEEE 14th International Workshop on Advanced Motion Control (AMC), pages 612–617.

United Nations Environment Programme (2015). Solid waste management.

Wang, L., Li, W., and Li, P. (2019). A comprehensive survey of vision-based human action recognition methods. Sensors, 19(5):1005.

Wang, W., Zhang, X., Li, J., and Liu, J. (2021). Garbage detection algorithm based on deep learning. In 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pages 958–962.

Wang, Y., Zhao, W. J., Xu, J., and Hong, R. (2020). Recyclable waste identification using cnn image recognition and gaussian clustering. arXiv preprint arXiv:2011.01353.

Yu, Y. (2020). A computer vision based detection system for trash bins identification during trash classification. Journal of Physics: Conference Series, 1617(1):012015.

Zhang, Y. et al. (2023). An image-based human-robot collision avoidance scheme. Journal of Computational Design and Engineering.