Game of Cat and Mouse

Crista Falk, Vanalata Bulusu

Abstract

For stereotypically finicky pets, like cats, interest in a new toy is not guaranteed, nor is prolonged play. Interactive toys, famously the self-moving ball Cheerble, seek to address this issue by offering a "smart" solution to engaging one's pet. However, no cat is the same and one toy's "smart" behavior does not account for the variety of play styles cats can exhibit. Moreover, a constant complaint of these toys are that they constantly run into walls, furniture, and even right into the cat itself. Thus, we propose a new interactive cat toy which utilizes "cat-in-the-loop" feedback to personalize its engagement and evasion behavior to suits each cat's long-term enrichment needs. In this project, we explore the dynamics at play in "cat and mouse"-style interactions, in which one agent–the cat–aims to capture, and the other–the (toy) mouse–intends to tease while avoiding capture. Using a virtual 2D simulation, we trained a mouse agent to interact with a cat agent, engaging the cat's interest without terminating the play session by getting caught or colliding with obstacles.

1 Introduction

Chronic animal boredom is a widely observed phenomenon, occurring for creatures living in nature and in captivity [2]. The risk of animal boredom can pose a distressing conundrum for pet owners, especially for those who work away from home and may be absent for hours at a time each day. Chronic boredom is known to co-occur with depression and can leave pets susceptible to other adverse health outcomes. For this reason, toys are crucial to ensuring a good quality of life for pets.

Popular cat toys that exist on the market, such as the Cheerble Ball and Giociv Interactive cat ball, perform random movements but are not attuned to a particular cat's behavior. Cats are adapted to hunting in the wild and enjoy toys that mimic a good 'cat and mouse chase' [4]. Prey, like mice make directed movements to avoid getting caught by the cat. Cat toys make random, not directed movements, sometimes making them very easy to catch or too difficult. Over time they become predictable and the cat becomes bored or frustrated. We want to build a cat toy that learns a particular cat's behavior and becomes more challenging the more the cat plays with it. Thereby, keeping the cat en-

gaged.

In this project we sought to simulate the interaction between a cat agent and moving toy using a gridworld environment to better understand the "cat and mouse" dynamic underlying engaging play.

2 Background Related Work

Autonomous robots, especially the Rumba cleaning robot, are familiar to many households. Rumbas use infrared sensors and ultrasonic sensors to detect distances from walls and furniture [7]. We plan to use a similar combination of detection devices to mark the distances between our toy and the environment. Rumbas construct static maps of each owner's house and then optimize paths to avoid obstacles. However, this method is not suitable for a cat and mouse chase where the obstacle is a fast, dynamic cat's paw [1].

Traditional autonomous bots that use static maps are not very good at handling dynamic obstacle spaces [1]. Several environments like hospitals or the road have continuously changing environments that are difficult to map. Solving the problem of dynamic obstacles could be very helpful for assistant bots or cleaning bots for the roads. Previous work on dynamic obstacle spaces have utilized Q-learning [5, 3]. Q-learning does not depend on the behavior of the agent (off-policy) and does not need a model of the environment to learn an optimal policy, making it particularly suited to avoiding dynamic obstacles.

With a flexible algorithm like Q-learning, designing good state spaces and actions becomes paramount. Previous work with autonomous bots have used states such as orientation of the agent, and distance from an obstacle. Actions used were velocity, accelerate, decelerate, stop, turn left and turn right [1]. To make the computation on high-dimensional spaces faster, an RNN was used to predict Q-values [3, 9]. In future works, we would plan to implement a neural net on top of our Q-learning algorithm to allow our toy to quickly adapt to a cat's play patterns.

Unfortunately, after weeks of struggling with hardware and electronics blunders, we have opted for a digital approach to this problem, reducing the intended state space from one with several sensors (such as ultrasonic and infrared collision detectors). Instead we model obstacles and walls in the form of numerical values in the gridworld and dynamics of the cat and mouse in the form of turn-taking, making independent updates to one's velocities of travel.

3 Technical Approach / Methodology / Theoretical Framework

3.1 Model Overview

In the system, we incentivize the mouse with high reward for proximity to the cat and minimal reward for movement otherwise,. We incentivize the cat to be near mouse with high reward as well (with minimal negative cost otherwise, simulating expended energy due to movement). Colliding with the wall or any obstacle results in negative reward for the mouse. All rewards are sampled from a Gaussian distribution to account for real-world variation due to noise. We end an episode upon successful cat-contact with the arm's tip, indicating the mouse of the duo has been "caught."

3.2 Environment

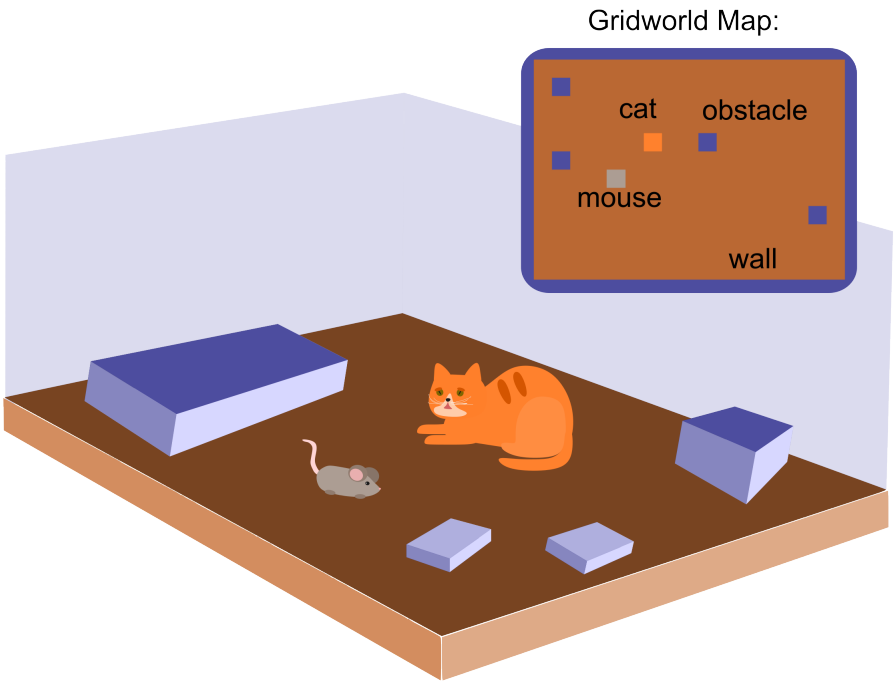

Figure 1: Representation of the cat agent, mouse agent, and obstacles in grid-space environment.

3.3 Algorithm

3.3.1 Model Specification

Much of the effort in this project went into crafting reward functions which might encourage the behavior we desire.

- States (separate for each agent):
  $C_{cat\ or\ mouse}$ = Cat or mouse's position in room,
  $V_{cat\ or\ mouse}$ = Cat or mouse's horizontal and vertical velocities

  $$S_{c,v} \leftarrow ([c_x, c_y], [v_x, v_y]); \forall \{cat, mouse\}$$

  $$\text{C, V} \in \mathbb{Z} \in \text{given room configuration}$$

  Velocity is constrained to (-2,2)

- Actions (same options for each agent):
  $A_{cat,mouse}$ = Changes to the horizontal and vertical velocities

  $$A_a \leftarrow v_x + a_x, v_y + a_y;$$

  $$a_x, a_y \in (-1, 0, 1)$$

- Cat Energy: We factor in cat tiredness (e.g., boredom), which after exceeding a threshold, will force the episode into its terminal state. Cat tiredness increases once per turn, and the rate increased during cat inactivity/decreased during engaged play (e.g., proximity to the mouse).

- Costs:
  $R_{t|mouse} \leftarrow \mathcal{N}(-5, 1)$ for hitting an obstacle or wall $R_{t|mouse} \leftarrow \mathcal{N}(-10, 5)$ for sedentariness (no movement) $R_{t|cat} \leftarrow \mathcal{N}(-5, 1)$ for sedentariness (no movement)

- Rewards:
  $R_{t|cat} \leftarrow \mathcal{N}(5, 1)$ each turn until terminal state (cat catches mouse) since cats love zoomies
  $R_{t|mouse} \leftarrow \mathcal{N}(1, 0.5)$ each turn until terminal state (cat catches mouse)
  $R_{t|cat,\ mouse} \leftarrow \mathcal{N}(8, 3), \mathcal{N}(5, 2)$ when in a 1 block proximity to other agent (i.e., the other agent is within 8 grid squares surrounding the agent's position after taking an action)

3.3.2 Pseudocode

Initialize, for S states and A actions:

$Q(s, a) \leftarrow 0$; $R(s, a) \leftarrow 0$

$\forall A : Policy(s) \leftarrow 0$

Loop for n episodes:
$s, a, r \leftarrow$ Perform run of agent in environment
$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma max(Q'(s',a') - Q(s,a)]$

if $a = argmax_a(Q(S))$ **then**
    $Policy(S) \leftarrow (1 - \epsilon) + \frac{\epsilon}{n(A)}$
else
    $Policy(S) \leftarrow \frac{\epsilon}{n(A)}$
end if

---

4 Experimental Results / Technical Demonstration

4.1 Evaluation Criteria

Because engagement is the name of the game here, we opted toward a duration based approach for evaluating the system's success. The longer the toy engaged the cat before an episode terminates, the higher the reward accumulated.

4.2 Performance Evaluation Results

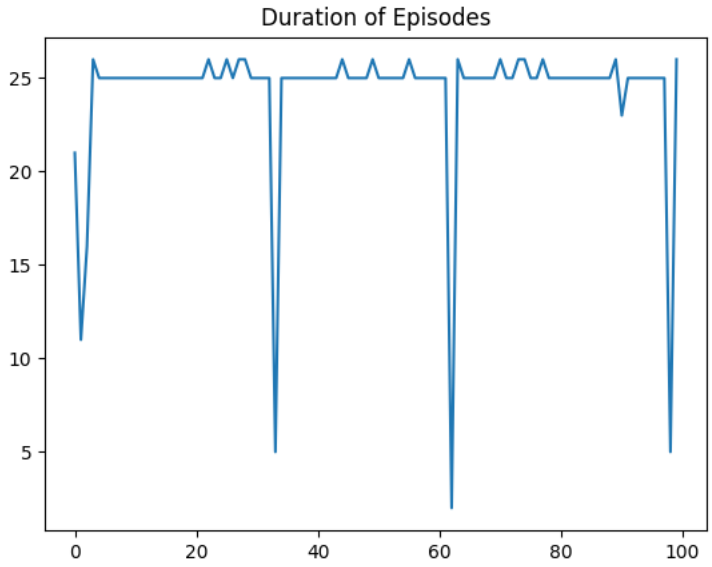

Figure 2: Duration of episodes tended to hover at the max, given the constraint of cat tiredness, which we considered a relative success despite occasional early termination due to hitting obstacles.

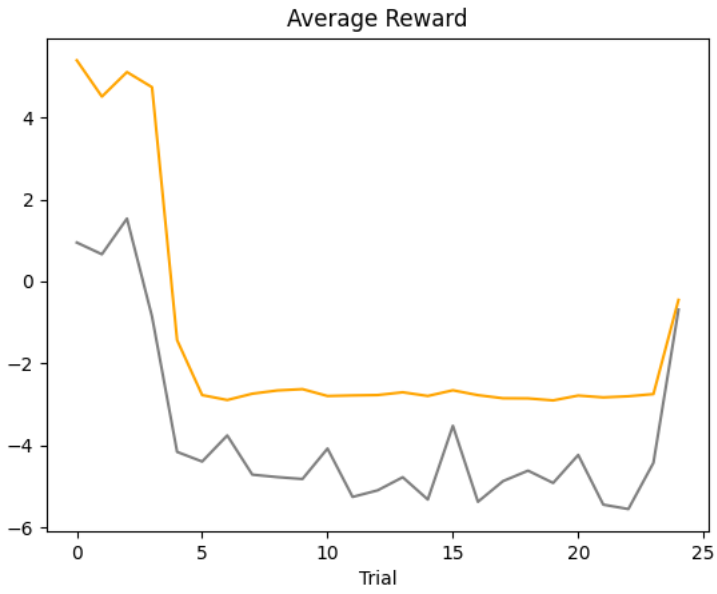

Figure 3: The average reward decreased initially but leveled out as trials continued, when averaged over episodes, showing signs of increase toward the end.

5 Conclusion and Future Work

5.1 Summary of Accomplishments

We successfully trained two agents to avoid obstacles and move in proximity to one another, furthering our aim of generating cat and mouse play behavior. We used q learning to establish dynamic agent behavior in a randomly generated environment of obstacles, mimicking a typical living room or cat play area.[1] The reward results were less than ideal, but with further tuning, we hope to improve this. The fact that reward did not continue to decline monotonically, however, is a positive sign.

5.2 Limitations and Future

One limitation of the project is the lack of realism in the simulated, grid-world environment. Discrediting the dynamics of the cat's movement behavior and kinematics in the world might miss out on crucial details of the system. Moreover, having to instantiate two agents led to many assumptions regarding the cat model which would not be present in our desired cat-in-the-loop final product.

[1] A video of the training is publicly available at the Cat And Mouse repository

For this reason a future goal is to simulate both the cat for this project in a more realistic setting. We will seek out software enabling us to program robotic RL agents in continuous code tasks using code alone: MuJoCo [8]. MuJoCo (Multi-Joint dynamics with Contact) is an open-source physics engine aimed toward developers which is largely compatible with RL techniques [10, 6]. MuJoCo can be used for model-based computational approaches to problems using control synthesis, state estimation, system identification, and mechanism design with compatibility for Python3 programming.

Notably, OpenAI has created a Gym environment which allows the developer to work directly with the MuJoCo engine. Further, there is an openly available repository called MuJoCo Menagerie with pre-configured XML files of multi-joint robots to insert into our environments. The ability to implement many model instances with high levels of separation between data and model was an attractive feature of this tool for implementing our design in an environment with three dimensional and realistic body physics.

References

[1] Khawla Almazrouei, Ibrahim Kamel, and Tamer Rabie. Dynamic obstacle avoidance and path planning through reinforcement learning. *Applied Sciences*, 13(14), 2023.

[2] Charlotte C. Burn. Bestial boredom: a biological perspective on animal boredom and suggestions for its scientific investigation. *Animal Behaviour*, 130:141–151, 2017.

[3] Jaewan Choi, Geonhee Lee, and Chibum Lee. Reinforcement learning-based dynamic obstacle avoidance and integration of path planning. *Intelligent Service Robotics*, 14:663–677, 2021.

[4] Mikel Delgado. Play behavior in cats. *Clinical handbook of feline behavior medicine*, pages 46–63, 2022.

[5] Liwei Huang, Hong Qu, Mingsheng Fu, and Wu Deng. Reinforcement learning for mobile robot obstacle avoidance under dynamic environments. In Xin Geng and Byeong-Ho Kang, editors, *PRICAI 2018: Trends in Artificial Intelligence*, pages 441–453, Cham, 2018. Springer International Publishing.

[6] Vaddadi Sai Rahul and Debajyoti Chakraborty. Exploring reinforcement learning techniques for discrete and continuous control tasks in the mujoco environment. *arXiv*, 2023.

[7] Daniel Paul Romero-Martí, José Ignacio Núnez-Varela, Carlos Soubervielle-Montalvo, and Alfredo Orozco-de-la Paz. Navigation and path planning

using reinforcement learning for a roomba robot. In *2016 XVIII congreso Mexicano de robotica*, pages 1–5. IEEE, 2016.

[8] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[9] Xuesu Xiao, Bo Liu, Garrett Warnell, and Peter Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, 2022.

[10] Xiaohang Yang Zhiyuan Zhao Lei Zhuang Zichun Xu, Yuntao Li and Jingdong Zhao. Open-source reinforcement learning environments implemented in mujoco with franka manipulator. *arXiv*, 2024.