Cat Bot, an Automatic, Robotic Cat Toy

Lyn Grimes[#1], Ryan Kelley[#2], Leila Moran[#3]

[#]*Computer Science Department, North Carolina State University, Raleigh, NC, United States of America*

[1]fmgrimes@ncsu.edu.

[2]rgkelly@ncsu.edu

[3]lmmoran@ncsu.edu

***Abstract*—To their human owners, cats provide companionship and affection; to cats, human owners provide food, shelter, and, similarly, affection. Aside from caring for the basic needs of a cat (feeding, cleaning the litter box, etc), humans also have the responsibility of keeping the cat entertained, especially in the case of indoor cats, who may suffer from the lack of stimulation. Unfortunately, not all cat owners always have time to play with their cats. Robotics can be utilized to help alleviate this endless need for play without requiring the presence of the owner.**

I. Introduction

Domesticated cats have been a common pet for human beings throughout much of our history. As of 2024, 32.1% of United States households have a pet cat [1]. A responsible cat owner not only cares for the basic needs of their cats, but also for their emotional and physical well-being, and play fulfills these needs. Unfortunately, not all cat owners may have the time to always keep their cats busy—inevitably, there are times where a cat owner's attention cannot be with their cat.

Traditional cat toys, such as feather dusters, are effective at occupying the attention of cats; however, they require manual operation by a human. This is not inherently a problem given a human owner with enough free time, but often a cat's preferred amount of play time is far greater than the human's feasible amount of free time. Our approach involves combining a traditionally effective cat toy with robotic automation to effectively entertain a cat while simultaneously leaving a human owner free to perform other tasks.

To this end, our group has created a robot that detects a nearby cat and begins moving a cat toy to entertain this cat. A more detailed description of the robot, as well as a discussion of individual team member contributions can be found in the Technical Approach section.

II. Related Work

Research into existing work on robotic cat toys was done to inform the design of Cat Bot. Some of this existing work also led to research into animal welfare as a whole, informing our design and testing process, and identifying some areas where our robot fails.

Relevant to our work is the paper discussing Autonomous Feline Entertainment Robot (AFER) [4]. This paper contains discussion regarding animal wellness in relation to the field of Animal Computer Interaction (ACI), and how to best design a robot with these considerations in mind. This paper was the beginning of a somewhat long rabbit hole into ACI that also informed our process of testing. Special emphasis was put into ensuring our test subject had salient ways to withdraw from testing [3]. Ultimately though, our design fails to be truly animal centered. The problem we set out to solve is human centered, having to do with their time spent away from their cats.

Research into human centered approaches to the robotic entertainment of cats was also done. One design in this research detailed an internet connected system that had future plans to implement an AI to control the robot's behavior [6]. The paper addresses the privacy concerns this would bring, and, along with other factors, influenced our group's decision to train an offline AI model rather than feeding data from our robot to some online model where images from within someone's household may be exposed.

III. Technical Approach

The basic overview of how the Cat Bot robot works is as follows. Two sensors, one camera, and

one LIDAR, are used to detect a cat. Every 30 seconds, the LIDAR system scans the environment for a few moments. When testing, this was every 5 seconds. A motion detection algorithm analyzes this collection of LIDAR scans to determine if motion has occurred surrounding the robot. If motion is detected, the camera takes a picture. This picture is passed to an AI model that will determine whether or not a cat is in the picture. If a cat is in the picture, the motor on top of the robot begins moving the attached cat toy.

A. Materials List

- Raspberry Pi 4. 4GB, ARM 64-bit, running Ubuntu 24.04
- Logitech USB Camera
- Scanse Sweep LIDAR sensor
- 3d Printed Frame
- Motor
- Cat Toy
- Printer Paper
- Tape
- ROS 2 Jazzy

B. Machine Learning Model

Because the machine learning model was intended to be deployed on a Raspberry Pi, it needed to be both fast and lightweight and suited for image classification. In order to save time and avoid extensive hyperparameter tuning, Leila first chose to use a pre-trained model as a base model instead of building one completely from scratch. She chose mobileNetV2, a CNN (convolutional neural network) as our base model because it was designed to efficiently perform computer vision tasks on small devices (mobileNetV3 was designed for mobile phones).

Leila's initial attempts at training resulted in significant overfitting within the first three epochs, which she then mitigated by excluding the top layers of the base mobileNetV2 model and freezing the first 100 layers (the rest were left open for training). She later implemented dropout layers and early stopping to prevent further overfitting during training. Other models that were not based on mobileNetV2 and had larger datasets were also tried, but she found that her first attempt had the best performance due to its smaller size.

To source images for the model, she wrote a python script that pulled from Google's OpenImages V7 using FiftyOne, an open source tool for building datasets. Specifically, she used FiftyOne's Dataset Zoo library to download the images. 20,000 images were downloaded, with 10,000 being of cats and the other 10,000 of non-cat objects such as humans and furniture. The script divided those images into datasets (cat vs non-cats) with 80% for training, 10% for testing, and 10% for validation. The code is available on our repository here.

With binary cross entropy as the chosen loss function, predictions made by the model produce a value between 0 and 1, with values closer to 0 indicating the likelihood of a cat in the image ("There is a 0.001% chance that this is not a cat") while values closer to 1 indicate that the image is unlikely to have a cat ("There is a 99% certainty that this is not a cat.").

The original code for training the model is present in a python notebook on the project repository or here in this document. The model is incorporated into the robot here.

The model was made using tensorflow and open source images.

C. Motion Detection Algorithm

The motion detection algorithm is made to use a rotating lidar sensor to detect movement. The lidar starts out by getting a starting map of the environment by spinning. Then on each consecutive spin it compares it to the last one to determine if something moved around it. We do this through a

weighted map. Since the sensor doesn't get the exact angles for distances every time it spins we give each line a weight that the difference between scans has to cross. This is all done in the blowup() function. Named due to us blowing up the weight of the scan points.

```
def blowup(samp, old, b):
    x = samp.distance * math.cos(math.radians(samp.angle / 1000.0) )
    y = samp.distance * math.sin(math.radians(samp.angle / 1000.0))
    x1 = old[b - 1].distance * math.cos(math.radians(old[b-1].angle / 1000.0))
    y1 = old[b - 1].distance * math.sin(math.radians(old[b-1].angle / 1000.0))

    x2 = old[b].distance * math.cos(math.radians(old[b].angle / 1000.0) )
    y2 = old[b].distance * math.sin(math.radians(old[b].angle / 1000.0) )

    x11 = 0
    xx = x - x1
    yy = y - y1
    y11 = 0
    y21 = y2 - y1
    x21 = x2 - x1

    v1d = ((yy)**2 + (xx)**2)**(1/2)
    v1t = 0
    if xx != 0:
        v1t = math.atan(yy/xx)



    v2d = ((y21)**2 + (x21)**2)**(1/2)
    v2t = 0
    if x21 != 0:
        v2t = math.atan(y21/x21)

    proj = math.sin(v1t - v2t) * v1d

    if (proj > value):
        print(proj)
        return True
    return False
```

Fig. 1 The blowup function

Not only did we add weights to our map but we also made it so that a certain amount of movement was needed to trigger a call to the cat detection node. This was because of the sensor bugging out sometimes and giving bad values. So now we have it to where there needs to be 4 sensor values different in a row (approx 1000 per rotation) for the robot to detect movement.

```
if(blowup(samp, old, b)):
    out = samp
    d = d + 1
else:
    d = 0
    out = None
if d > 3:
    break
```

Fig. 2 Code sample showing the code that detects for four changed values

D. Individual Contributions

TABLE I
CONTRIBUTION TABLE

Contribution	**Members**
Camera Node	Ryan, Leila
Cat detection node	Leila, Ryan
Documentation (Reports, Presentations)	Lyn, Ryan, Leila
ML Model for Cat Detection	Leila
Motion Control	Lyn, Ryan
Motion Detection Node	Ryan
Part Acquisition	Leila, Ryan
ROS Environment Implementation	Leila, Lyn, Ryan
ROS Environment Setup	Lyn, Ryan
Testing & Debugging	Ryan, Leila, Lyn

E. Physical Design

Our first design of the robot was to have a moving bot that would seek out the cat. This was later re-designed to something stationary that would wave something in front of the cat with a motor on top.

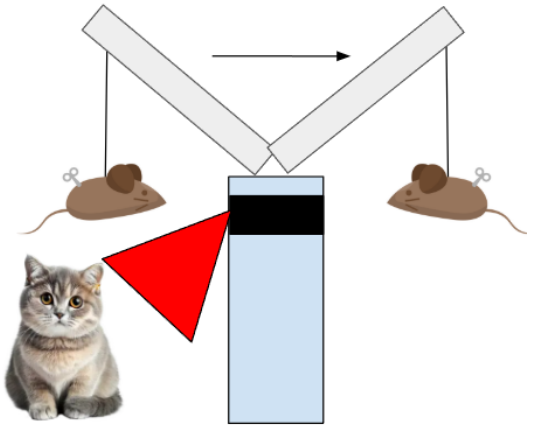

Fig. 2 initial mockup

The design was made to have a 360 degree detection range however for the scope of this paper we only made it detect movement in front of the robot. The design uses a lidar on the bottom and a set of cameras on the top to take pictures in the direction of movement to determine if the movement was a cat. The motor for the top was originally intended to be a servo motor so we could wave the arm directly in front of the cat using the angle, but due to a programming issue we were unable to implement this. Then the motor on top rotates the toy. We 3D printed the parts at the makerspace at NCSU.

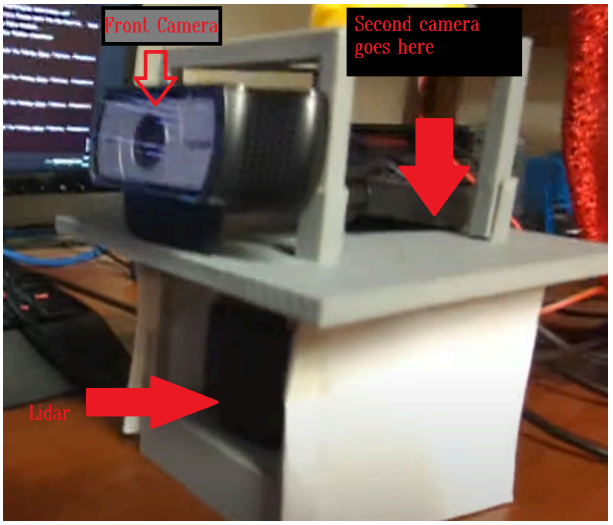

Fig. 3 The robot (sides are blocked so the lidar doesn't detect movement in them.)

F. Code Design

We used ros2 jazzy for this project and made nodes to make the implementation of the functions easier. The first node is the lidarListener which detects movement then calls the service check_for_cat. With the request the brain will forward the request to the cat_detector service and detection node. Then based on the result will either wait for another check_for_cat or send the request to the play_with_cat service and motorControl node. Motor control controls the motor and waves the toy in front of the cat.

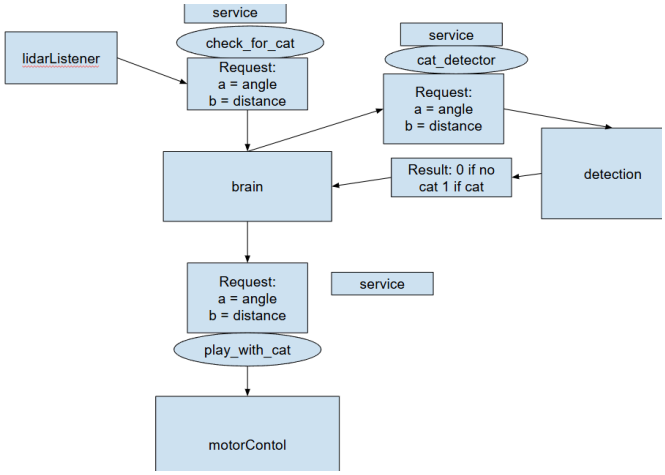

Fig. 4 Node map (results that don't matter are not included)

IV. Lessons Learned

A. Training the Machine Learning Model

See section Technical Approach B: Machine Learning Model for Leila's experiences

B. Simple Error in Motion Control Algorithm

When testing our motion detection algorithm, we were getting unexpected output. Whether or not motion was occurring did not correlate with whether or not the algorithm thought motion was occurring. When debugging, we saw that all the code was being reached and all the formulas were being used correctly, so we were unsure what could be wrong. Eventually, Ryan found the actual error, using math.degrees instead of math.radians. Math.degrees converts radians to degrees, whereas math.radians converts degrees to radians. Assuming the simple parts of our algorithm were correct and beginning with investigation of more complex pieces caused us to miss an easy to fix error.

```
def blowup(samp, old, b):
    x = samp.distance * math.cos(math.radians(samp.angle / 1000.0) )
    y = samp.distance * math.sin(math.radians(samp.angle / 1000.0))
    x1 = old[b - 1].distance * math.cos(math.radians(old[b-1].angle / 1000.0))
    y1 = old[b - 1].distance * math.sin(math.radians(old[b-1].angle / 1000.0))

    x2 = old[b].distance * math.cos(math.radians(old[b].angle / 1000.0) )
    y2 = old[b].distance * math.sin(math.radians(old[b].angle / 1000.0) )
```

Fig. 5 Variable setup for the motion detection algorithm using the correct math.py function

C. Servo Control in Ubuntu

When making the code to control the actuation of the robot, our research into potential libraries did not lead to easy servo control. In previous robotics projects, our group members used Arduino libraries to easily set servos to some angle [2]. For Ubuntu running on a raspberry pi, the ig libary was what we determined to be the best option [5]. This library only seemed to include Pulse-Width-Modulation (PMW) controls, meaning our servo control script would, rather than tell the servo to reach a position, would control how much power each pin in the servo would receive. With this limitation in mind, we decided to change to a simple motor with only

two pins. One for forward and one for reverse.

```python
class PlayWithCat(Node):

    def __init__(self):
        super().__init__('play_cat')
        self.srv = self.create_service(AddTwoInts, 'play_with_cat', self.play_callback)

    def play_callback(self, request, response):
        response.sum = request.a + request.b

        self.get_logger().info('Incoming request\na: %d b: %d' % (request.a, request.b))

        try:
            for i in range(3):
                lgpio.gpio_write(h, REVERSE, 0)
                lgpio.gpio_write(h, FORWARD, 1)
                time.sleep(1)
                lgpio.gpio_write(h, FORWARD, 0)
                lgpio.gpio_write(h, REVERSE, 1)

                time.sleep(1)
        except KeyboardInterrupt:
            # Stop sending power to both pins
            lgpio.gpio_write(h, REVERSE, 0)
            lgpio.gpio_write(h, FORWARD, 0)
            lgpio.gpiochip_close(h)
        lgpio.gpio_write(h, REVERSE, 0)
        lgpio.gpio_write(h, FORWARD, 0)
        return response
```

Fig. 6 Final motor control code that causes the motor to do a pre programmed action when it receives a request through the play_with_cat service. Control C can be used to stop the motors manually for testing purposes.

D. ROS2 Python Library Interaction

ROS2 has some weird interactions with python libraries that we came across when developing the robot. To run the cv2 python library you first need to open the camera then take the image. Usually this is done when first running the program because of the overhead cost of opening the camera. Normally this works out fine, the camera is only accessed when it needs to be and takes photos in real time. This however did not work with our robot. The camera was only used when calling the service and client. Somehow its interaction with ROS made it so the camera was always taking photos after the first picture. This resulted in every consecutive call to the service taking off the stack of photos taken while other nodes were running and not the real time camera. So even when the cat walked away it would still be stuck with photos of the cat that were taken while the motor was running. We fixed this problem by re-opening and closing the camera every time the cat_detector service was called.

```python
def detect_cat(self, request, response):
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    image_path = "temp.jpg"

    cv2.imwrite(image_path, frame)
    #time.sleep(5)
    cap.release()

    # Resize image
    img = image.load_img(image_path, target_size=(224, 224))

    # convert to array
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0  # Normalize to [0, 1]
```

Open Camera

Close Camera

Fig. 7 Code that opens and closes the camera after each call to the cat_detector service

V. Conclusions

Our team set out to create a robot that can fulfill a cat's need for play when a human is unavailable. To do this we designed a robot that would automatically use a traditional cat toy when a cat was nearby. We did successfully create a robot that moves a cat toy when a cat is nearby; however, this robot fails to solve our stated problem in a few ways.

The cat was not observed playing with the toy while it was in motion. When testing the bot, the cat played with the attached cat toy even when the Cat Bot was not moving the toy, but never when the robot was moving the toy. This implies that the cat toy alone was more entertaining to the cat than the Cat Bot system as a whole.

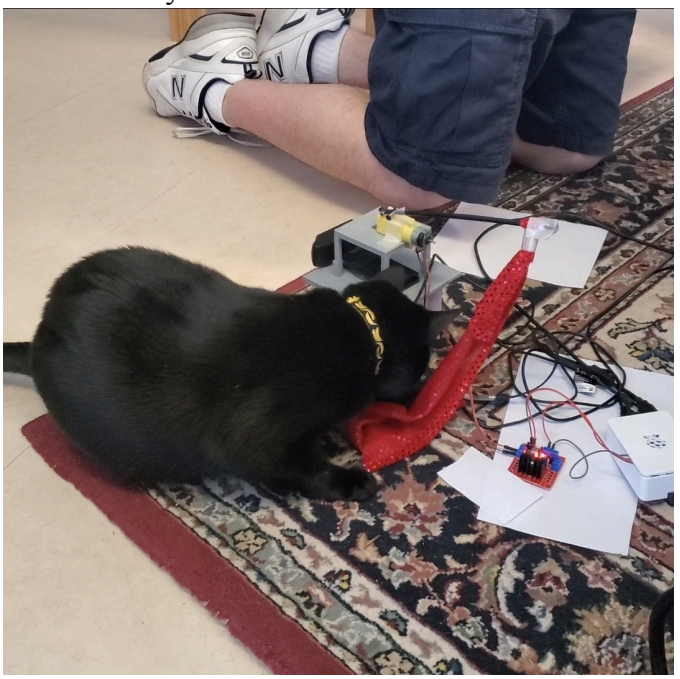

Fig. 8 Freya playing with one of her cat toys attached to the stationary Cat Bot.

The image analysis model can, in some circumstances, believe that a dog is a cat and, along with approval from the motion detection algorithm, cause the Cat Bot to begin moving the toy. This does not prevent the device from playing with a cat, but the device is not tested for dogs and unexpected issues could arise in, for example, a household with both dogs and cats. Future iterations of Cat Bot should train the AI model against dogs in addition to training it to detect cats.

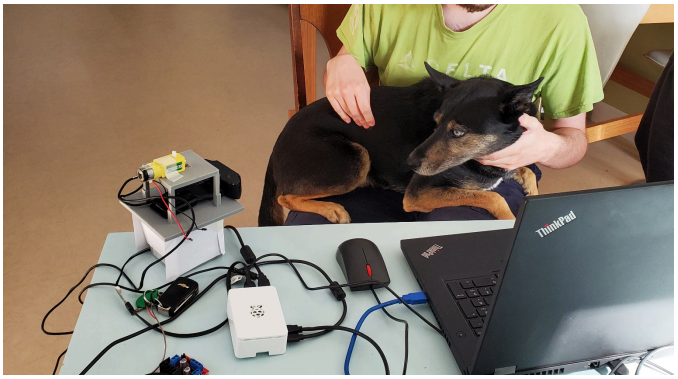

Fig. 9 Cat Bot detects a cat when a dog is in its view

The robot cannot be operated safely without supervision. All wiring of the present robot is exposed, as shown in Fig. 8, and poses a risk of injury if bitten into by a cat. For this reason, all testing had to be done with a team member bodyguarding the wiring and moving the cat away if she attempted to bite or claw the wires. Future versions of the Cat Bot should ensure that the cat cannot access the wiring.

This last issue does not have to do with the ways in which our robot fails to solve our stated problem, but is an important consideration for future work on Cat Bot. The device was not designed in a user centric way. Ideally, minimal human interaction is needed for the device to function. Therefore, the user of the device is the cat. Designing such a device to solve a human-centric problem is not designing the robot for its users. Any future work done on the Cat Bot should be animal centered for this reason.

References

[1] American Veterinary Medical Association, "U.S. pet ownership statistics," American Veterinary Medical Association, 2024. https://www.avma.org/resources-tools/reports-statistics/us-pet-ownership-statistics

[2] *Arduino.cc*, 2024. https://docs.arduino.cc/libraries/servo/

[3] C. Mancini, "Towards an animal-centred ethics for Animal–Computer Interaction," International Journal of Human-Computer Studies, vol. 98, pp. 221–233, Feb. 2017, doi: https://doi.org/10.1016/j.ijhcs.2016.04.008.

[4] C. S. Adams and S. M. Mizanoor Rahman, "Design and Development of an Autonomous Feline Entertainment Robot (AFER) for Studying Animal-Robot Interactions," SoutheastCon 2021, Atlanta, GA, USA, 2021, pp. 1-8, https://doi.org/10.1109/SoutheastCon45413.2021.9401864

[5] "lg library," Abyz.me.uk, 2020. https://abyz.me.uk/lg/py_lgpio.html

[6] Isla Xi Han and Sarah Witzman. 2023. Internet of Robotic Cat Toys to Deepen Bond and Elevate Mood. In Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction (HRI '23). Association for Computing Machinery, New York, NY, USA, 791–794. https://doi.org/10.1145/3568294.3580183

[7] U.S. Bureau of Labor Statistics, "The Employment Situation," Bureau Of Labor Statistics, Feb. 2025. Available: https://www.bls.gov/news.release/pdf/empsit.pdf