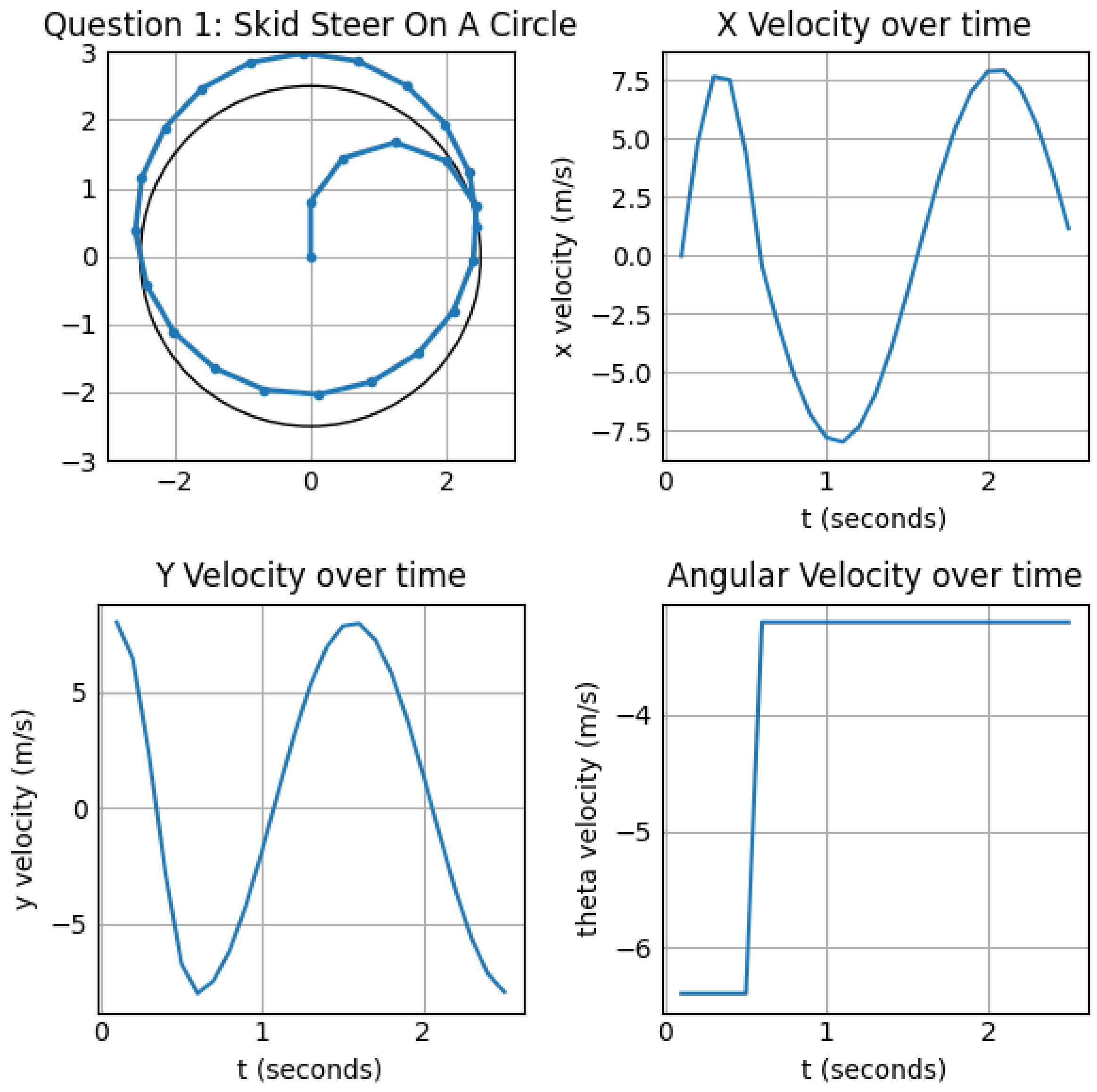
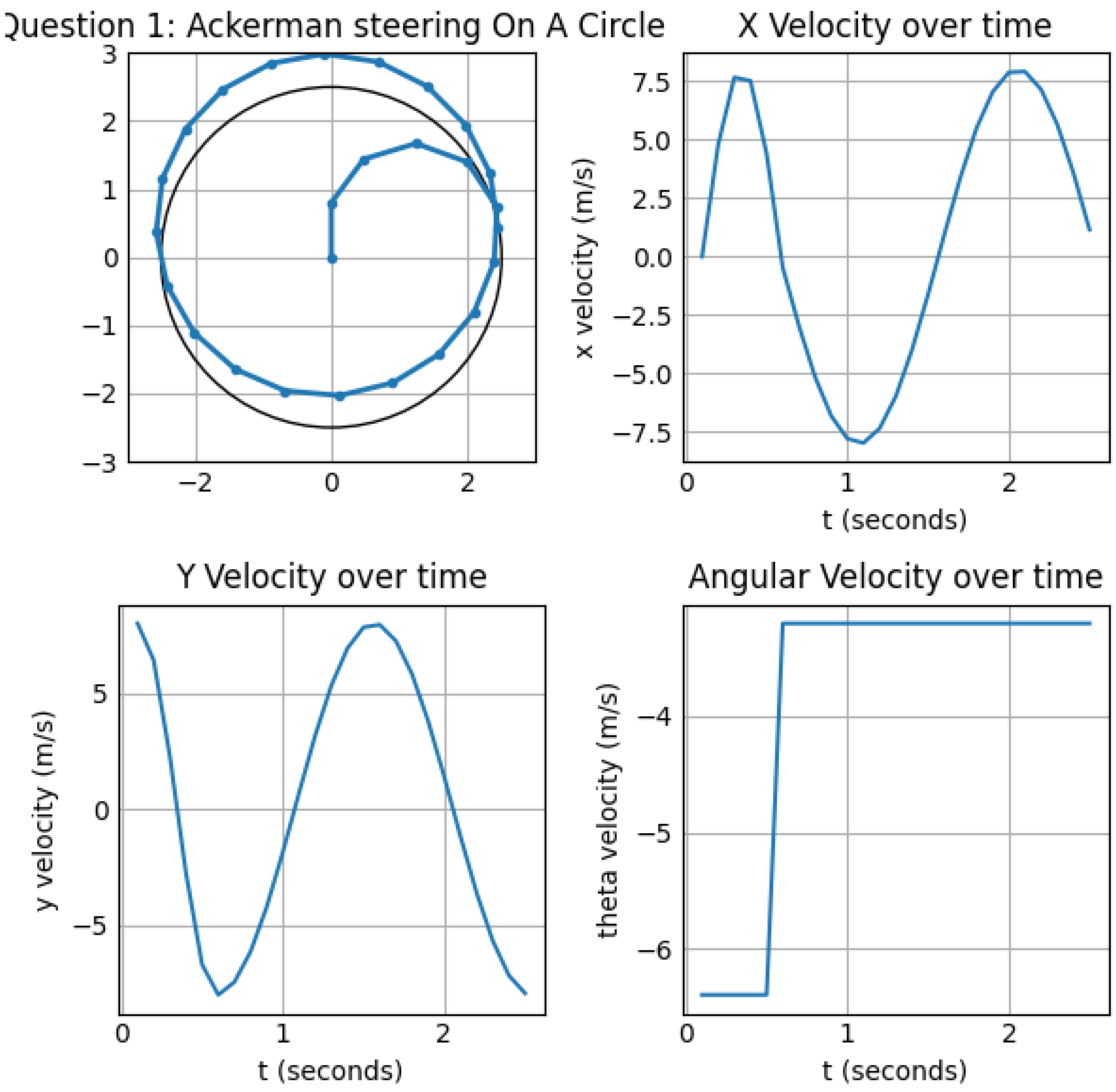
CS 5510 Robotic Intelligence
Midterm

Joshua Williams
Kyle Stimson
Sadie Atkinson
Trevor Jex

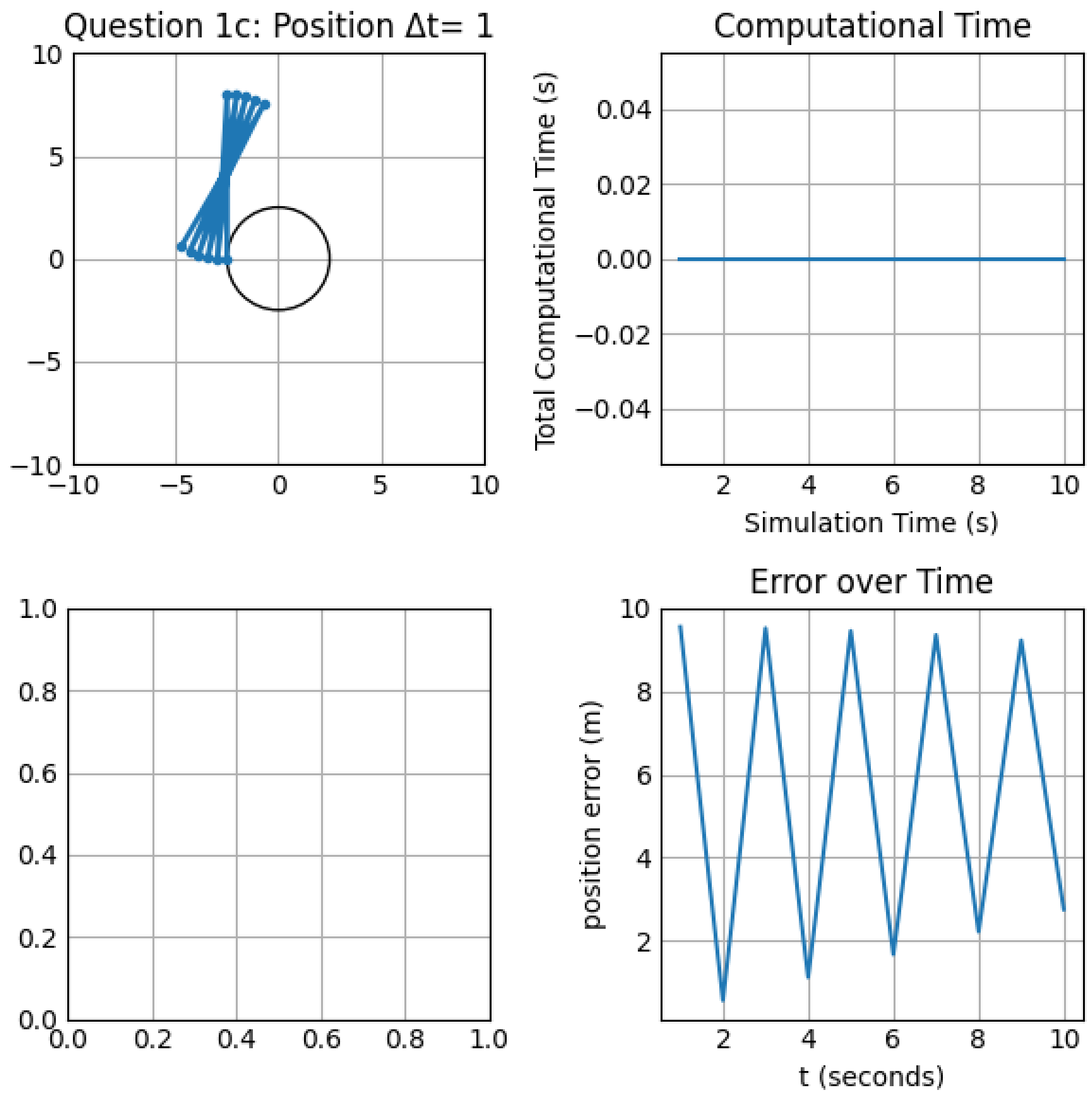
1. Moving in a car
   a. Design a skid steer robot to traverse a 5m circle, plot the path and trajectory.

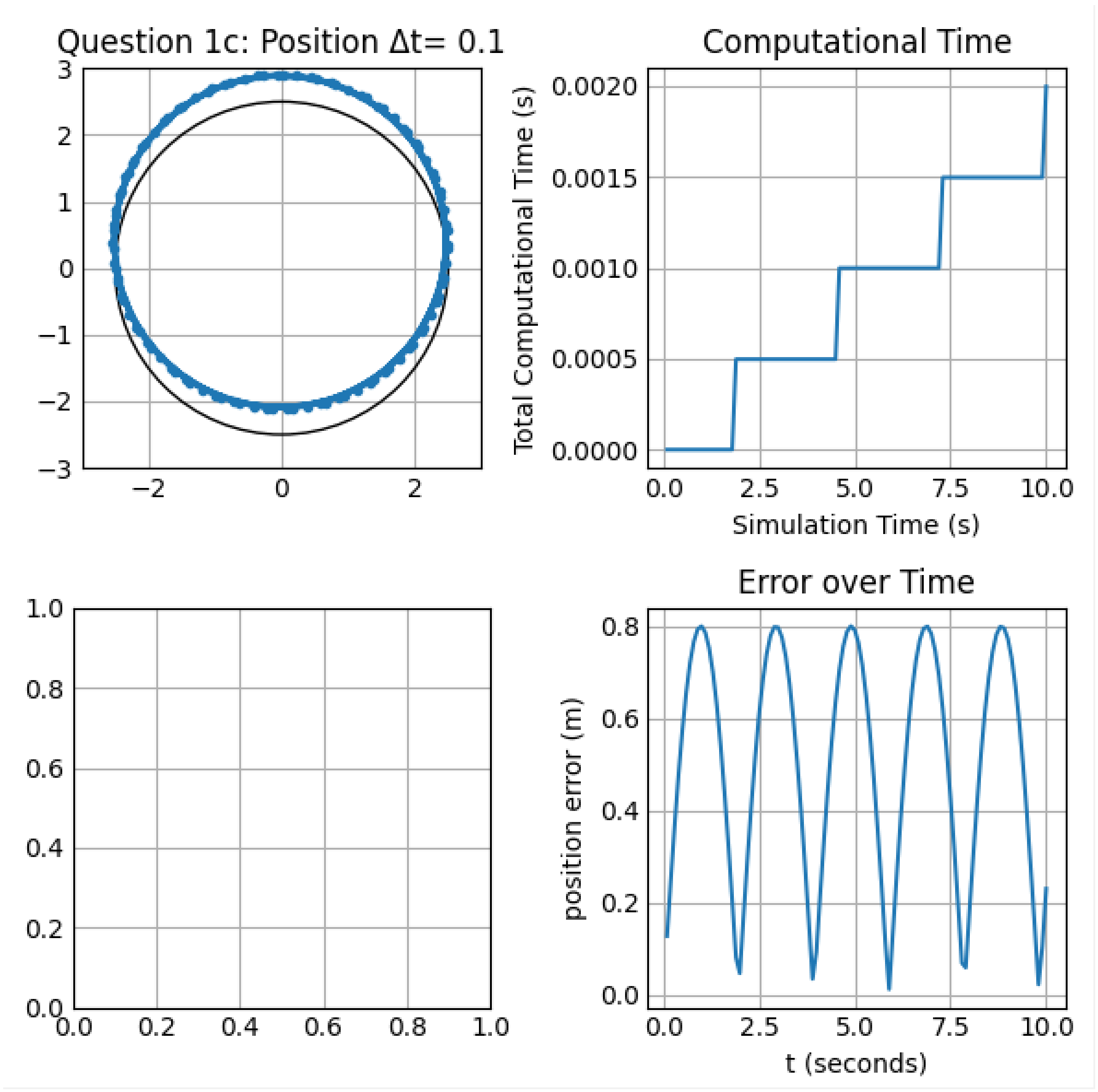


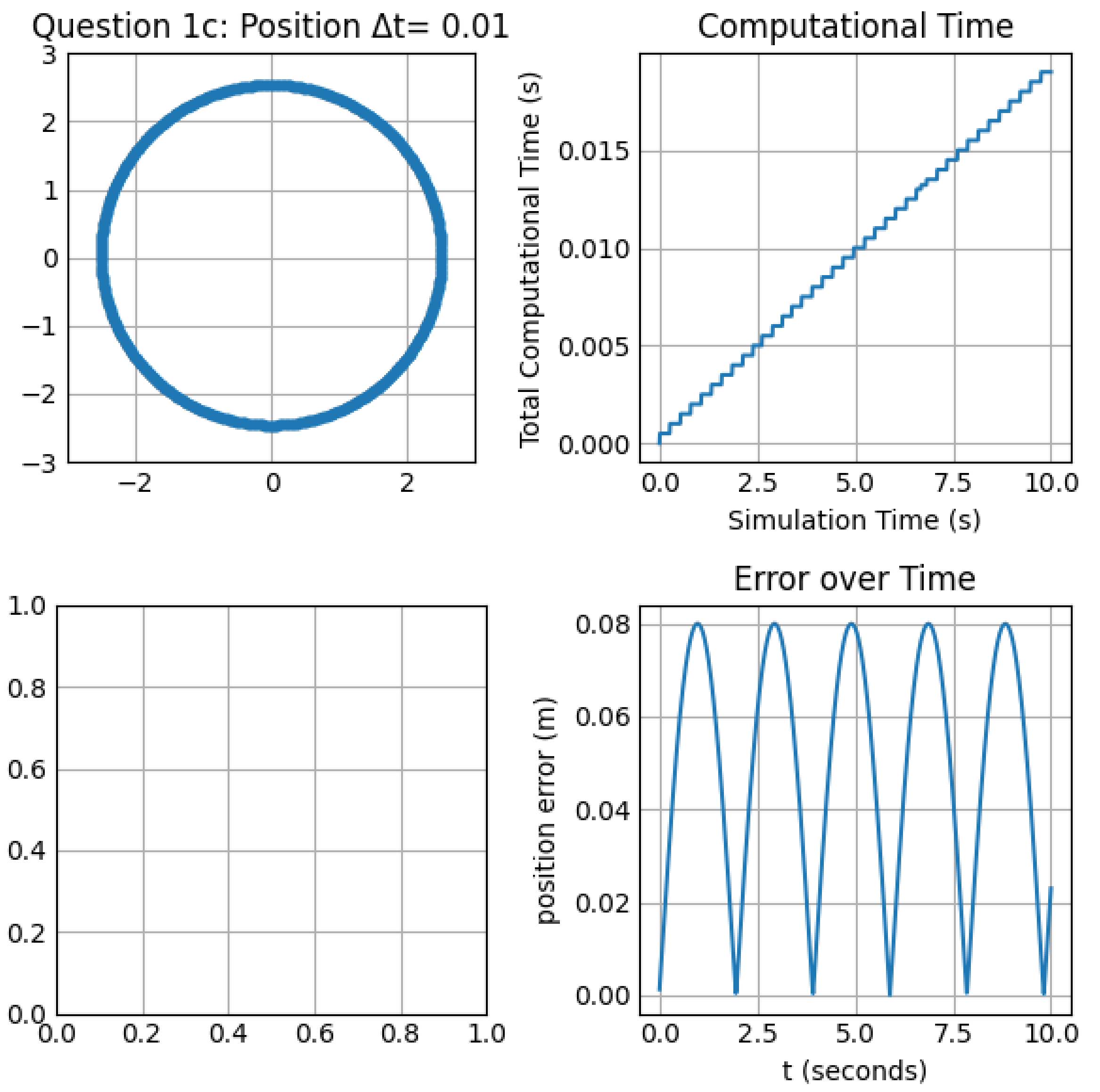
b. Traverse the same circle using Ackermann steering



c. Calculate the positional error of the Ackermann steering at Δt = 1,0.1,0.01

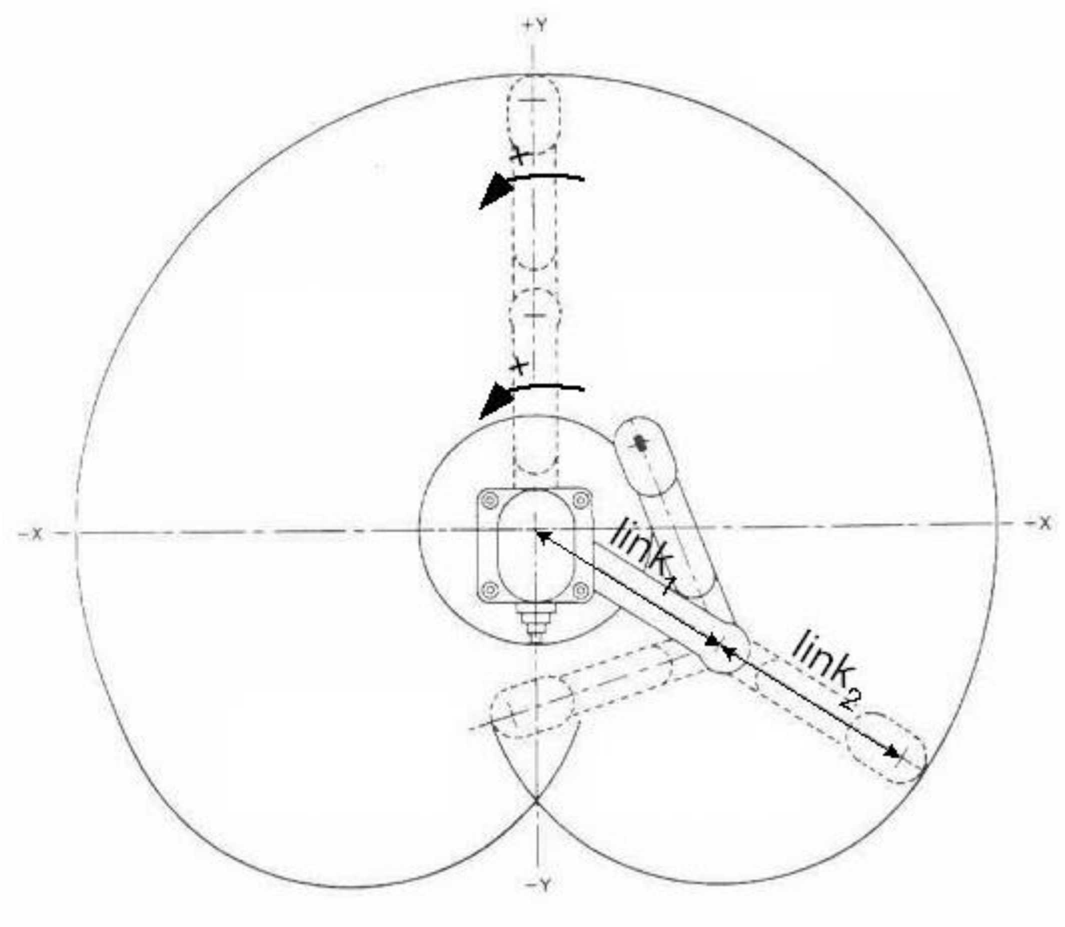



Question 1c: Position Δt= 0.1
Computational Time
Total Computational Time (s)
Simulation Time (s)
Error over Time
position error (m)
t (seconds)



2. SCARA Manipulators

a. Show the workspace for the robot



The farthest this robot can reach is 100 cm. The first arm cannot reach all the way around because of wires. However the second arm can bend to reach all around the base

b. DH params

Joints	Di(cm)	Ai(cm)	Ai-1(deg)	Θi(deg)
1	0	40	180	Θ1
2	0	40	180	Θ2
3	0	0	0	Θ3
4	D4	0	0	0

c. Position

x = 0.62314 m
y = 0.68637 m
z = -0.14 m

3. Inverse Kinematics with numerical approaches
    a. Compute joint angles and extension distances to get the robot to x=1.2, y=.8, z=.5
    X = 1.20030 Y = 0.80082 Z = 0.50612
    Θ 1 = -51.58178 Θ 4 = -101.91290 Θ 5 = 112.63239
    d2 = 0.40936 d3 = 1.43872

    b. What are the joint angles and extensions to get the same goal with an end effector at Θ1=0, d2=.2m, d3=.3m, Θ4=-90, Θ5=90, Θ6=40, d6=.2

       X = 1.20516 Y = 0.80367 Z = 0.50490
       theta 1 = -58.02727 theta 4 = -107.15793 theta 5 = 132.35992
       d2 = 0.36370 d3 = 1.58265

4. Balancing a pole
    a. Describe the equations of motion in the system

       There are two equations that are the secret of the cart and pole problem.

       The first one is angular acceleration - which depends on gravity, the force exerted, the current angle, the current angular velocity, and the mass of the cart and mass and length of the pole.

       The second one is x acceleration - which depends on the same things except gravity.

       As the pole falls further away from the center or its angular velocity increases, it requires a greater force in order to counter-act the falling. Also, if gravity is higher, or the mass of the pole or cart or length of the pole is larger, then a larger force will be required as well.

    b. Create code for a controller that would keep the pole balanced in air.
       From within the 5510IronBlimpsMidterm/ folder, run `python Q4/Q4b.py`

       Results:
       ```
       Failure to balance at
       T = 8.359999999999866
       Theta: 1.5892958554014158, 91.0599449121351degrees.
       Force: 0.04785968748806488
       Pos = -9.93713539054651
       ```

    c. What is the maximum angle that my pole can fall before it cannot recover if max Force F = 6n

       Results:
       ```
       unrecoverable angle = 0.1308996938995747
       Unrecoverable angle (degrees) = 7.499999999999999
       ```

5. Control and Reinforcement learning

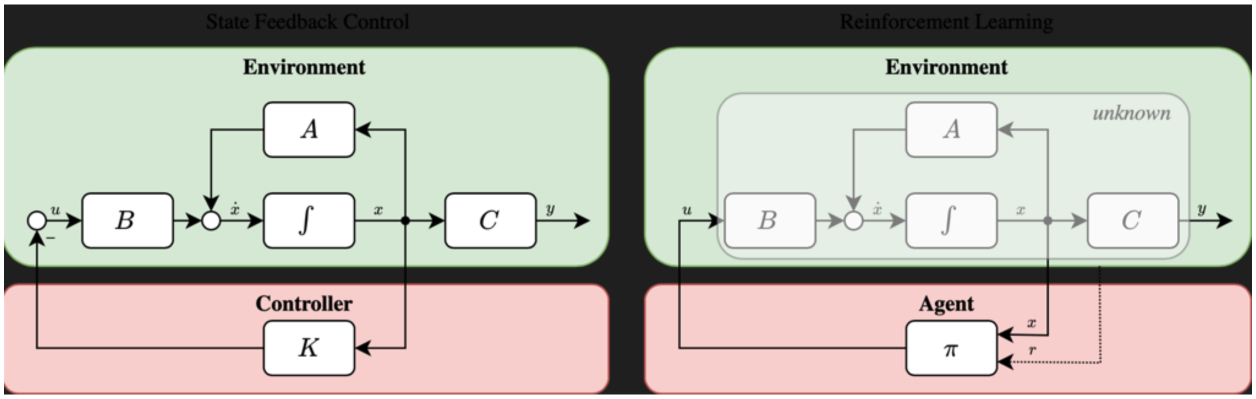
a. Explain three merits and three demerits of using reinforcement learning for mechatronic systems

Pros:
1. Good to solve complex problems with no known optimal solution.
2. Good when the only way to collect information about the environment is interacting with it.
3. Can learn to adapt a skill to a unseen but similar task. It could even adapt to hardware changes like wear or changing parts.

Cons:
1. Need a lot of data and computation which may be limited. High dimensionality may make models inefficient.
2. Wear and tear may be expensive when collecting training data.
3. Won't have direct control over the actions of the system, it could perform unexpected actions.

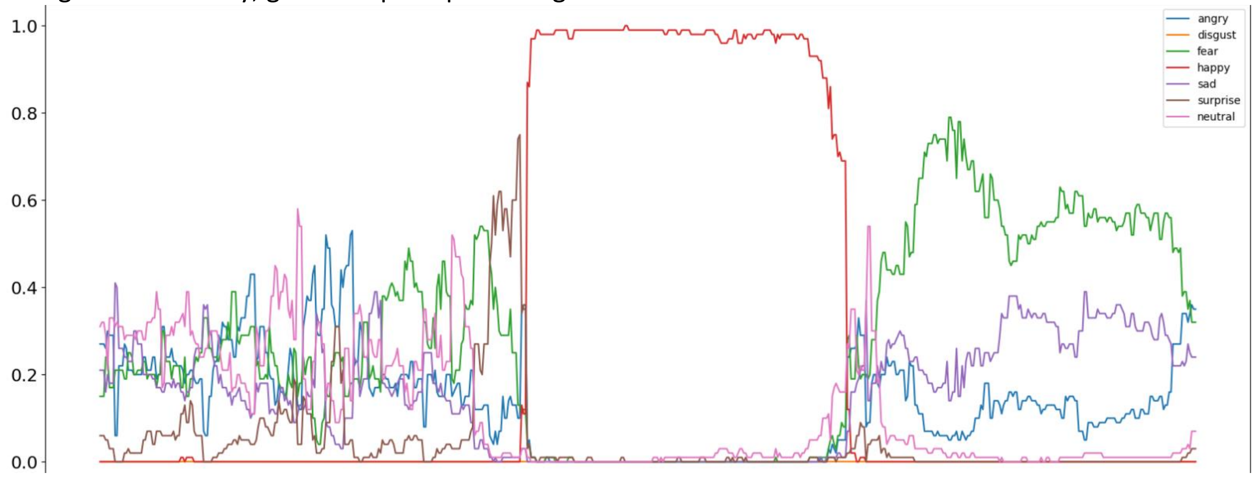
b. Draw a diagram for reinforcement learning and controls and contrast the two.

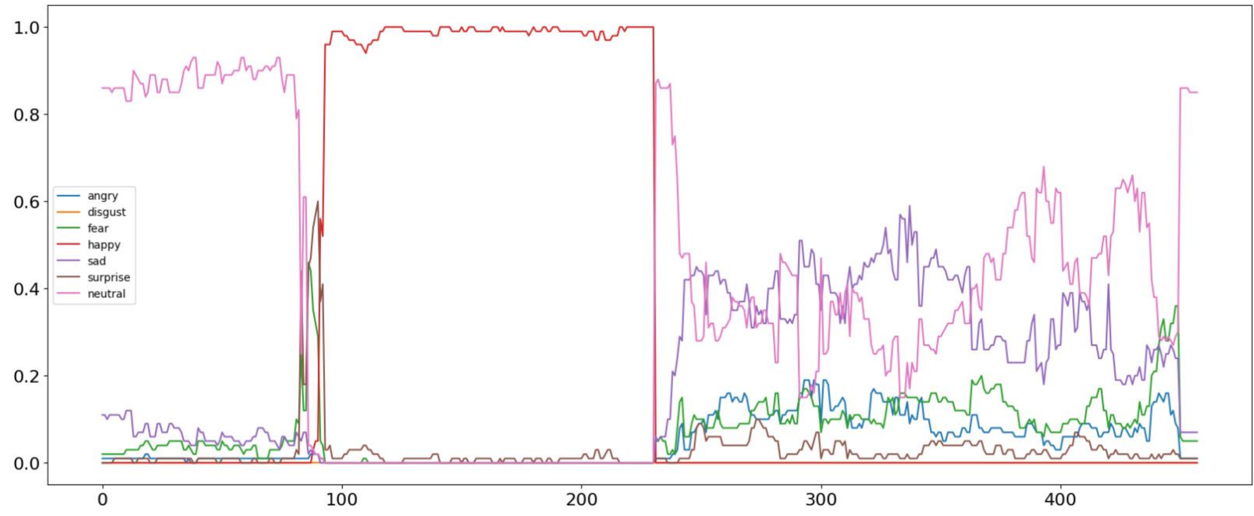
Both Controls and RL can sometimes be useful in similar situations to try to achieve the same results. They
both need observations from the environment and then adjust their future actions based on the observations.
However, they have many differences in how they actually work. Controls need to know the internal mechanisms
of the system and how they work, while RL does not. Controls will have a set of rules to follow depending on
the feedback it receives. It will react with a control for each scenario that it comes across. RL on the other
hand doesn't need to understand how the system works and can treat it as a black box. In this sense, it can
work in some scenarios where controls might not be able to. RL will reward correct tendencies based on the
results of previous actions performed and gradually improve. It will often require many more resources and
work to compute than a Controls model would need.



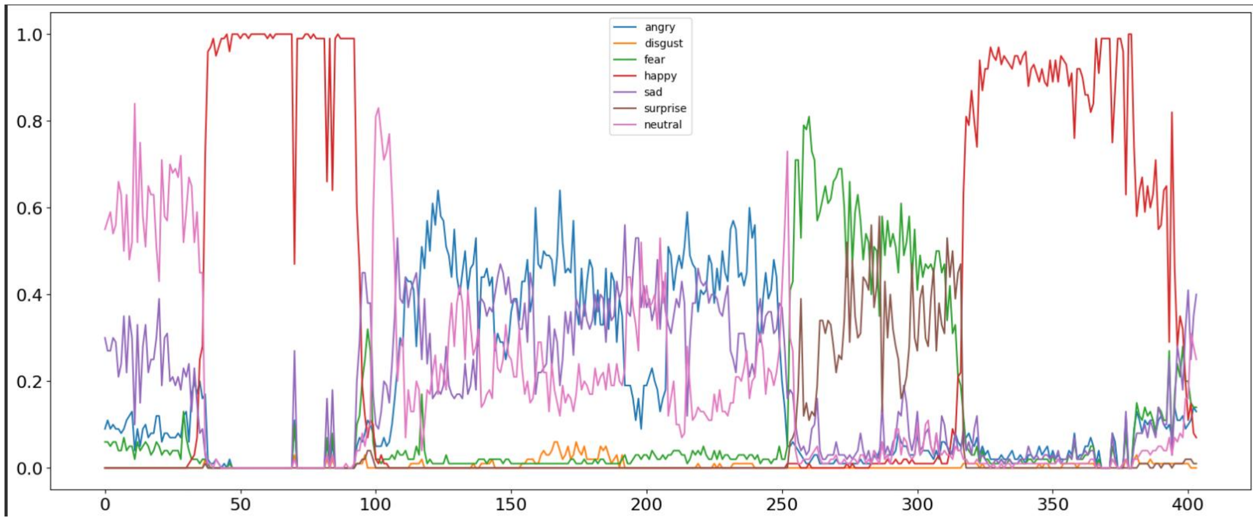
c. Use OpenAi Gym to create a trained reinforcement learning model for the cart and pole
   Train the model by running `python "Q5/Q5 train Q learn model.py"` from within the 5510IronBlimpsMidterm/ folder. Then, run `python Q5/Q5c.py`

6. Human emotions
   a. Using the FER library, generate plots predicting emotions over time for the noted videos





   b. Do the same with a webcam.
   Referencing Q6/Q6b_output.mp4



c. What are the logical applications for this using a robot? What about the ethical and legal consequences?

If a robot were able to accurately detect the emotions of humans, they could interact differently depending on the emotions that they see. This tool could have many different uses anytime that the robot is interacting with a human. If a tool was developed to use this in an interview process, an employer could use it to read how job candidates feel about different questions. Then they could decide to hire based on the responses. Or if it were used in education, they could try to judge how much the students were listening and understanding. There would probably be ethical concerns because of privacy. Some people would not like the idea of a robot trying to read the emotions on their face, especially if they didn't specifically know that it was happening. Ethical and legal decisions would have to be made concerning the use of this tool.

7. Motion Planning

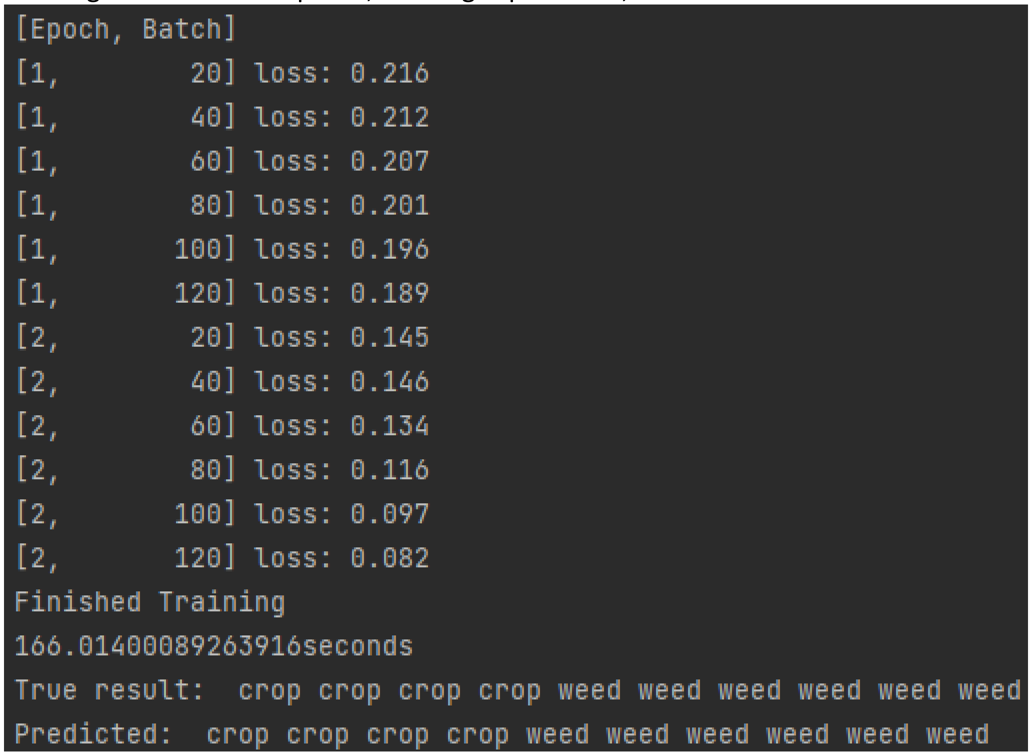
a. Implement an A* base planner, compare it with Djikstra, A-Star, RRT-Star, Bi-Directional A-Star, and BFS planners.

Algorithm	Average Path Cost	Average Time To Run
A* (our implementation)	101 nodes	0.497 seconds
A* (from AtsushiSakai/PythonRobotics)	85 nodes	0.746 seconds
BFS	85 nodes	0.623 seconds
Bidirectional A*	86 nodes	0.702 seconds
Dijkstra	100 nodes	0.589 seconds
RRT*	140.3 nodes	4.7846 seconds

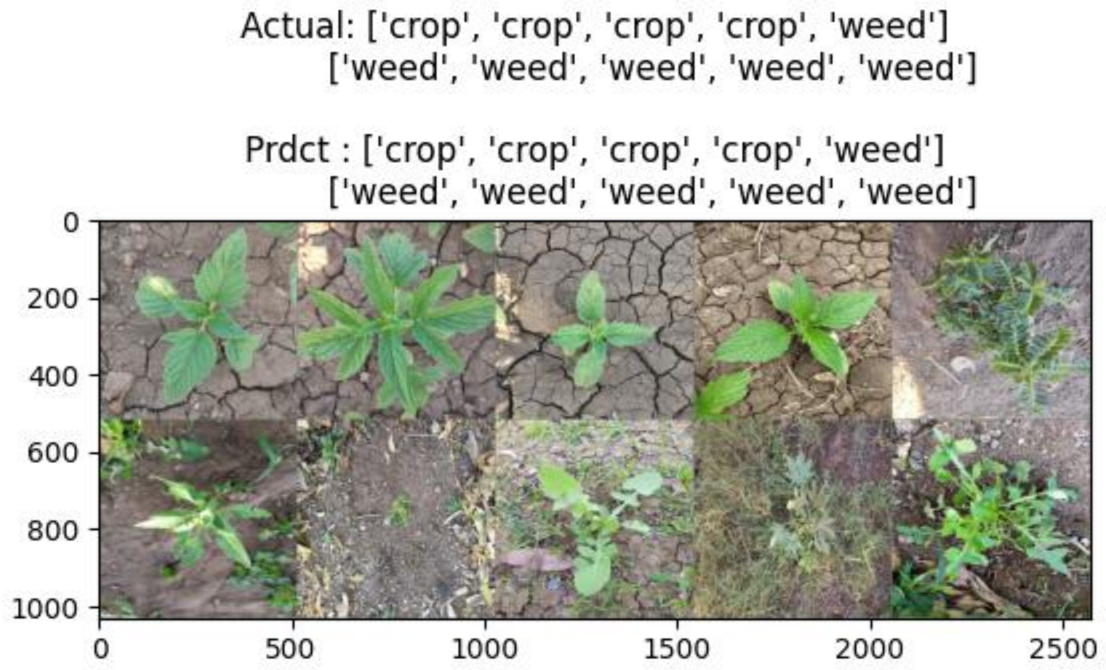
- The A* (not ours), BFS and Bidirectional A* provided the lowest cost path on average.
- The fastest algorithm was our implementation of A* which ran at an average of 0.05 seconds to converge.
- I would change our planner so that it would find slightly shorter paths. One of the ways we could do this is by modifying our cost function, or not having the algorithm finish as soon as it converges, but rather look for better paths.

- The "best" planner seems to be A* or Bidirectional A* since they find very good paths in a short-ish time. But it really depends on your use case. If quick paths are needed I'd use our implementation of A* even though it is slightly less efficient.

8. Object Detection
    a. Classify the 10 images from github.com/ravirajsinh45/Crop_and_weed_detection
    Utilizing a CNN for two epochs, 10 images per batch, and minibatches of size 20:



```
[Epoch, Batch]
[1,        20] loss: 0.216
[1,        40] loss: 0.212
[1,        60] loss: 0.207
[1,        80] loss: 0.201
[1,       100] loss: 0.196
[1,       120] loss: 0.189
[2,        20] loss: 0.145
[2,        40] loss: 0.146
[2,        60] loss: 0.134
[2,        80] loss: 0.116
[2,       100] loss: 0.097
[2,       120] loss: 0.082
Finished Training
166.01400089263916seconds
True result:  crop crop crop crop weed weed weed weed weed weed
Predicted:  crop crop crop crop weed weed weed weed weed weed
```



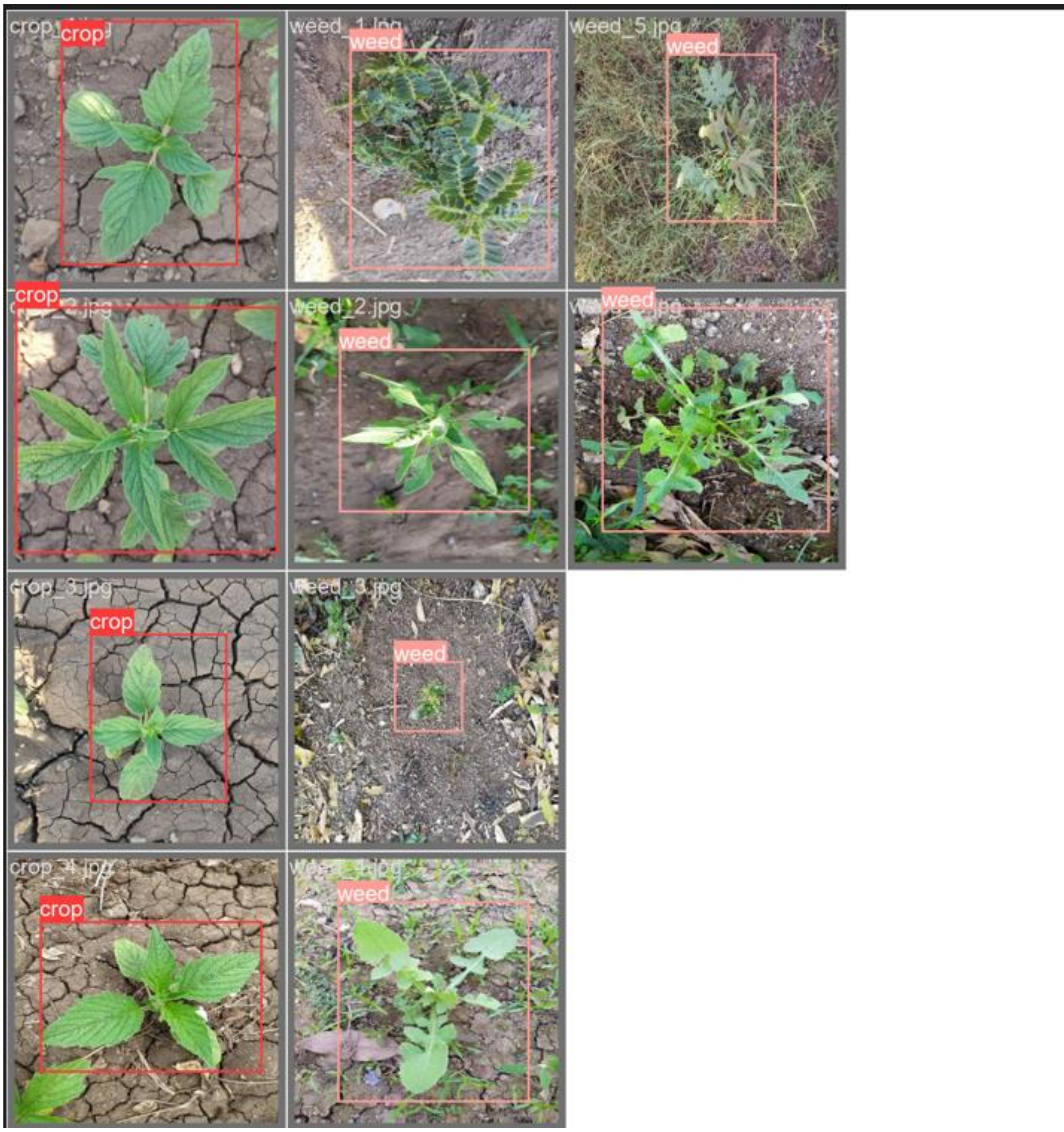
```
Actual: ['crop', 'crop', 'crop', 'crop', 'weed']
        ['weed', 'weed', 'weed', 'weed', 'weed']

Prdct : ['crop', 'crop', 'crop', 'crop', 'weed']
        ['weed', 'weed', 'weed', 'weed', 'weed']
```

b. Implement YOLO and note the differences.

The YOLO algorithm was extremely fast, much faster than running the CNN.
Upon testing, I was able to conduct dozens of epochs in the same timespan as it took me to run several on the CNN.

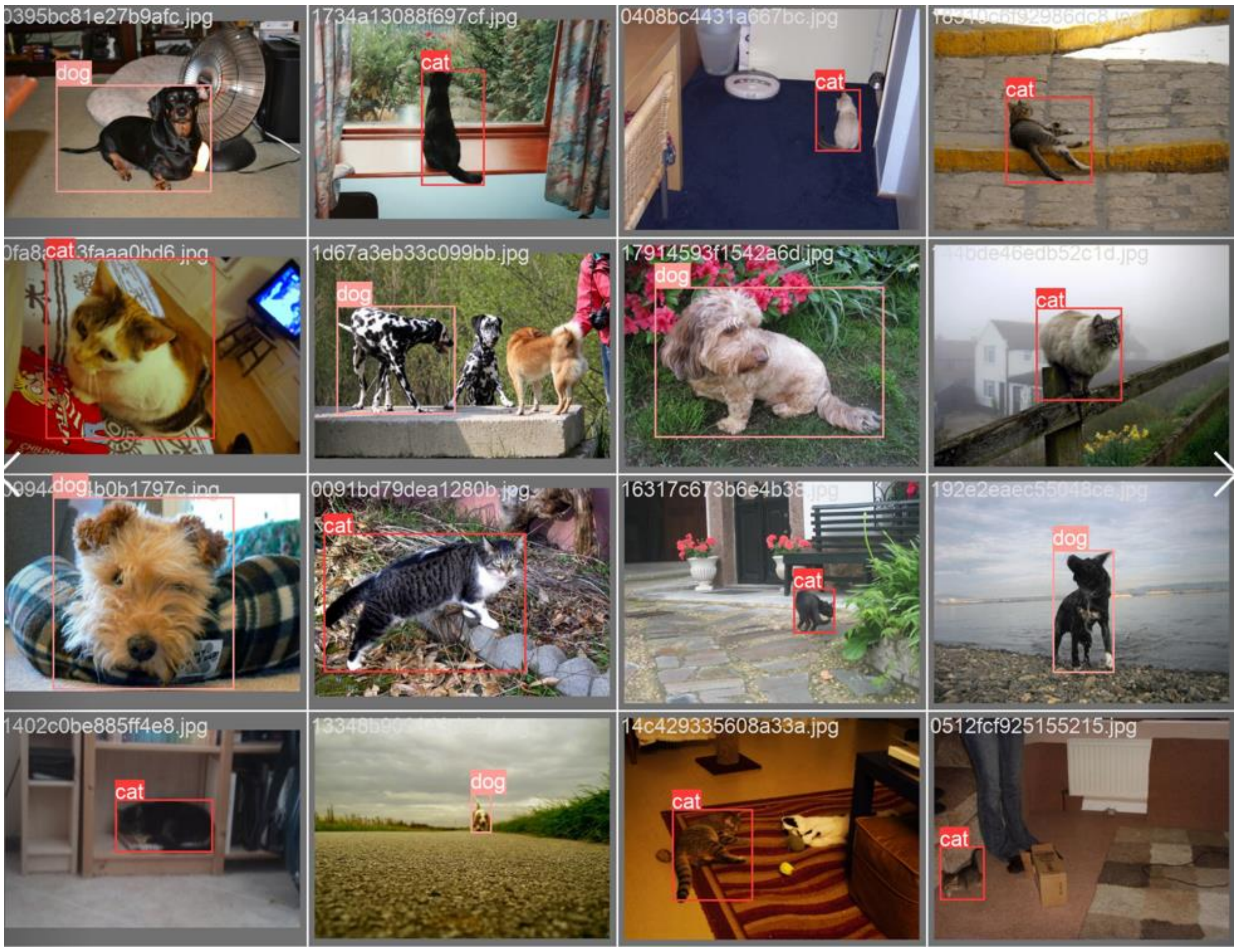
The CNN took a two stage approach to analyzing images, first detecting objects of interest within a given frame
And then moving through a second time to analyze each object and determine its identity.

On the other hand, Yolo passes through the image once to detect an object and analyze it at the same time. Additionally, YOLO had better accuracies quicker compared to the CNN.



c. Use Transfer Learning to pick an image classification algorithm and retrain it.

   Using a cats and dogs dataset, we used the pretrained weights of yolov5s.pt, and the yolov5s.yaml config froze the model, and ran the series of images to get the following results during validation.



9. Ethics of Robotics

# Question 9

## Part A

I think that a large part of robotics is technical, but there are definitely a lot of places where you get in muddy water ethically. If you aren't careful, you can start doing some really unethical things. We think that it's dangerous to consider robotics as purely technical, and that anyone involved in robotics should have some degree of ethical training/thinking.

## Part B

Isaac Asimov's 3 laws of robotics (simplified) are "do not harm human beings, whether through inaction or action", "obey human beings unless the order breaks the 1st law" and "preserve your own existence unless doing so breaks the 1st or 2nd law". Algorithmically this is super difficult to implement, because you have to get a computer to understand what "injuring a human
being" entails. You also have to build adequate communication systems so that the robot can process human orders whether through speech or otherwise. You also have to build a complex

hierarchy to be able to determine whether you protect your own existence, obey a human order or prevent harm to a human. There are so many different possibilities here that make it impossible to obey these 3 laws without having artificial intelligence that is far in the future.

However, you can build a simple hirerachy that will obey these laws partially - the robot may not be able to prevent all kinds of harm but you can program it with reasonable awareness of its surroundings. It also may not be able to respond to all types of commands but can have a purpose-built interface. It can also take reasonable measures to prevent known kinds of harm to itself.

The scenario we are choosing is a robotic train. Here is some basic pseudocode.

```
# if obstruction in path
  # brake at a reasonable speed to avoid collision
  # only brake up to a maximum deceleration - this is a precoded value (don't brake too hard
  # as to derail the train, but you can brake hard enough to permanently damage the brakes)
# else
  # if human says go
    # go, as long as speed is not above maximum safe speed
    # if going too fast would damage the train (but not harm people onboard), allow it but
    # warn the user
  # if human says stop
    # stop, as long as deceleration is not above maximum safe deceleration
    # if braking too fast would permanently damage the brakes (but not harm people
    # onboard), allow it but warn the user
  # if human says nothing
    # maintain speed, or slow down if the speed is above what is good for the train
```

This scenario prioritizes people's lives first, then human orders, and only then the wellbeing of the train itself.

## Part 3

I agree with both articles that the liability should rest primarily on the self-driving car manufacturer and not with the backup driver - but I do think that the backup driver has partial responsibility to make sure their car is in good working order. I think that if the car isn't in good working order and the backup driver is aware of that, then they should possibly be held liable. I think it would be good to have safety certifications that self-driving cars are required to pass to be able to be street-legal. Having a centralized safety certification would ensure that the requirements and expectations are clear, and shift blame off of the backup driver and onto the product manufacturer.

## Part 4

I think that laws in some cases (e.g. self driving cars above) are necessary for safety. I do

think that too much regulation could become a big problem, with companies (or individuals) manufacturing robots being bound by a lot of red tape, and unable to try out new things because
of laws. I think that too much regulation will especially harm individuals who are wanting to experiment with robotics.