Midterm CS5510

https://github.com/ShawnBraden/CS5510

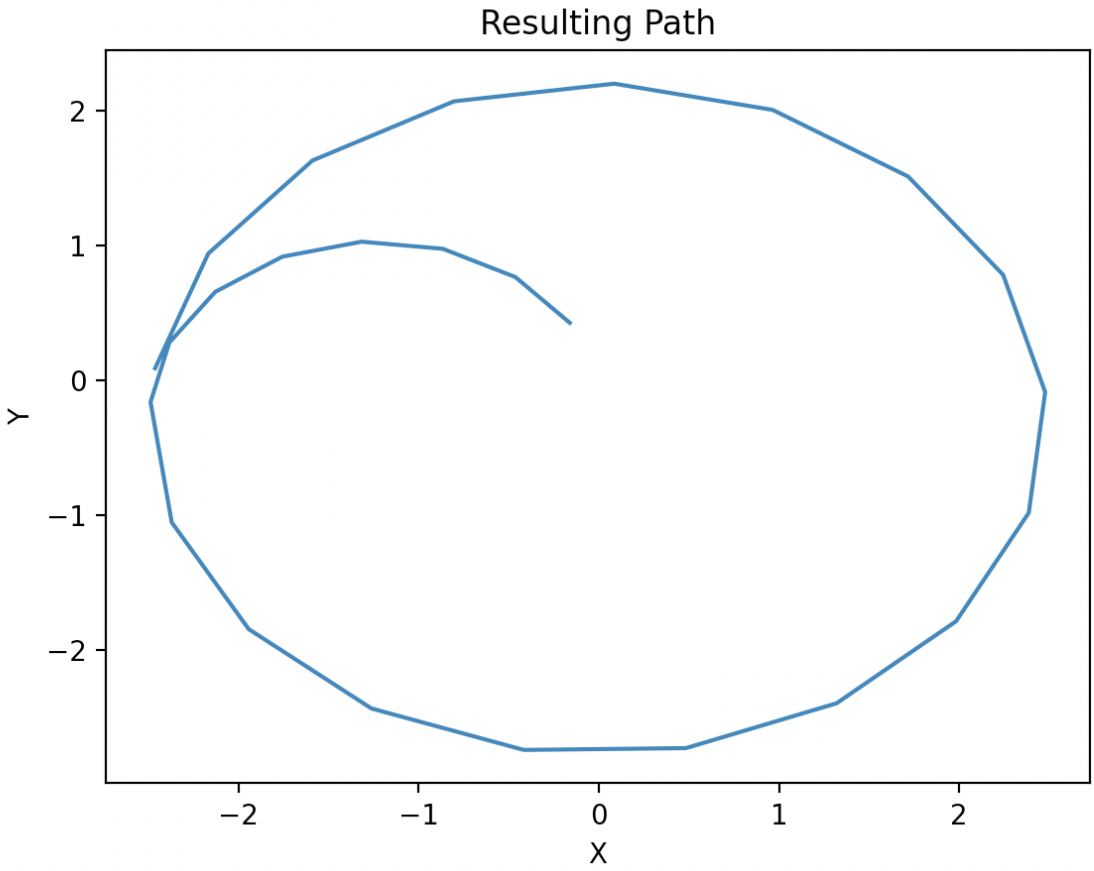
Problem 1

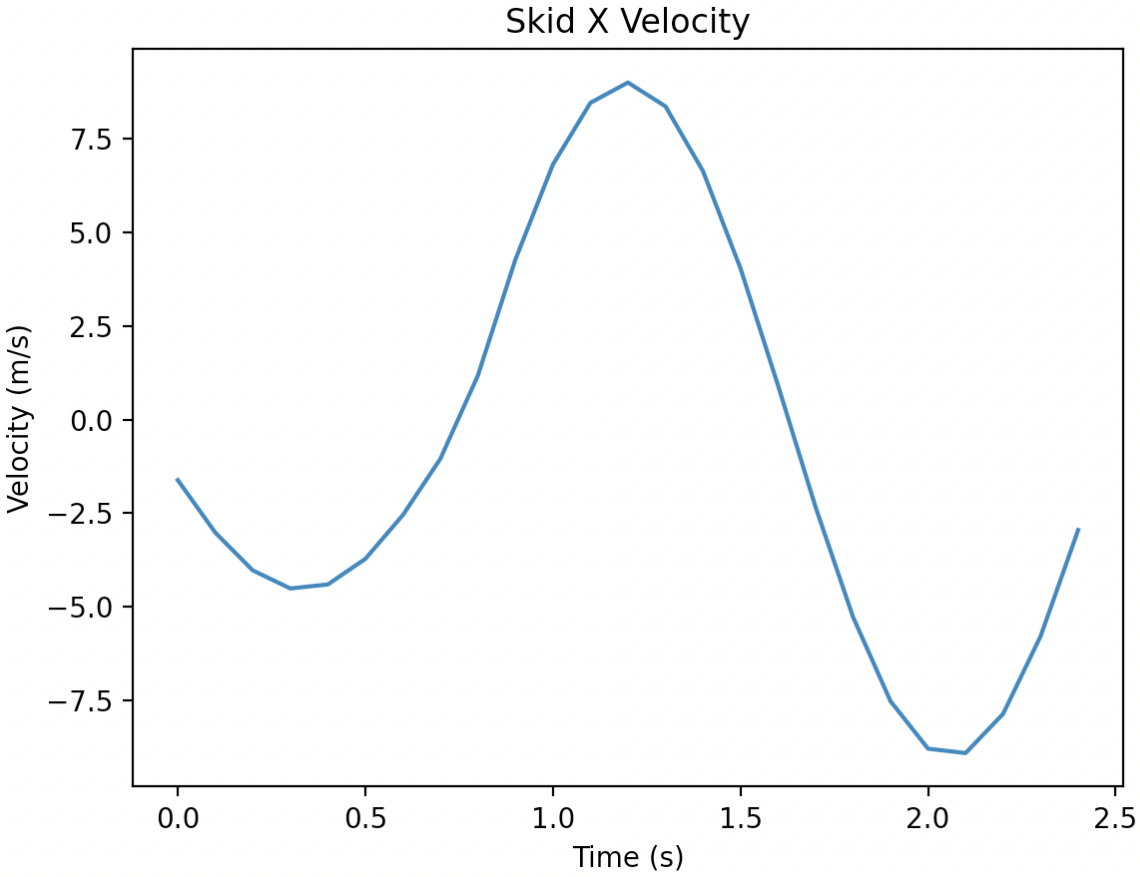
See midtermProblem1.py in our repo for the code.

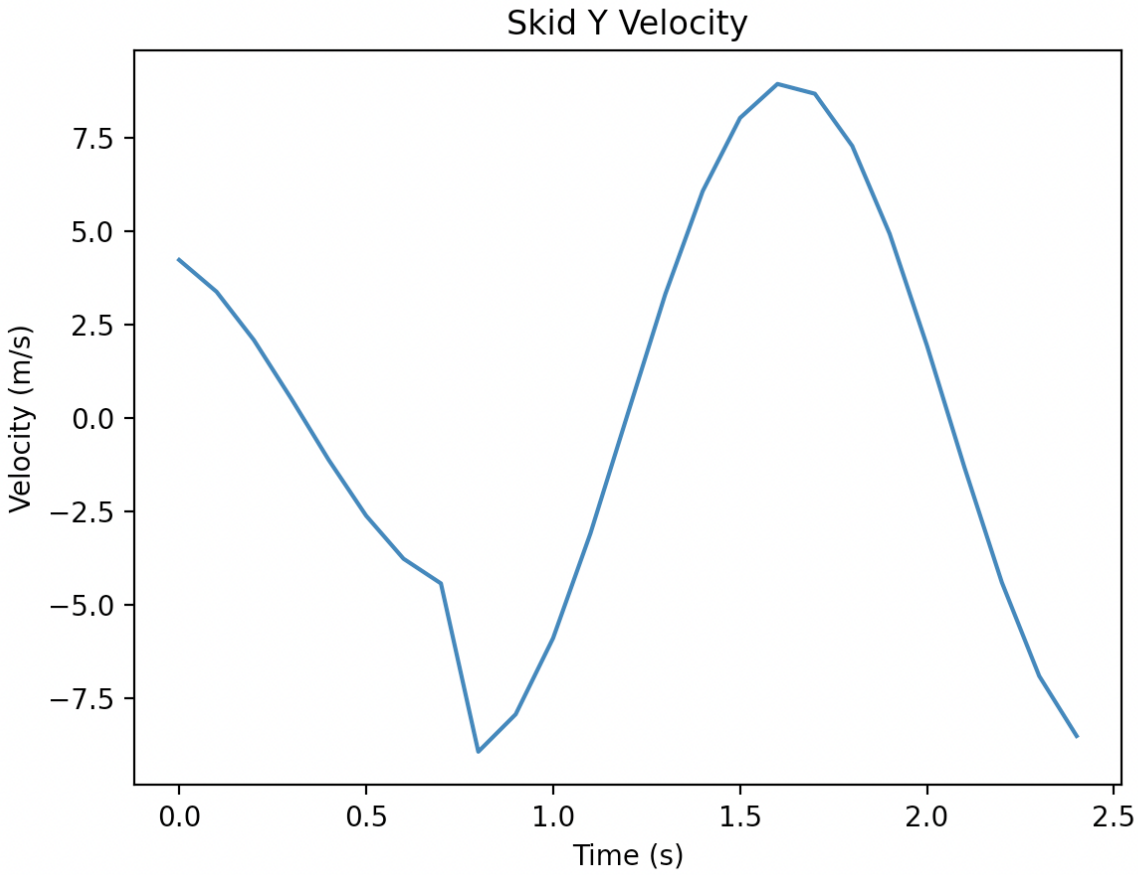
(a)

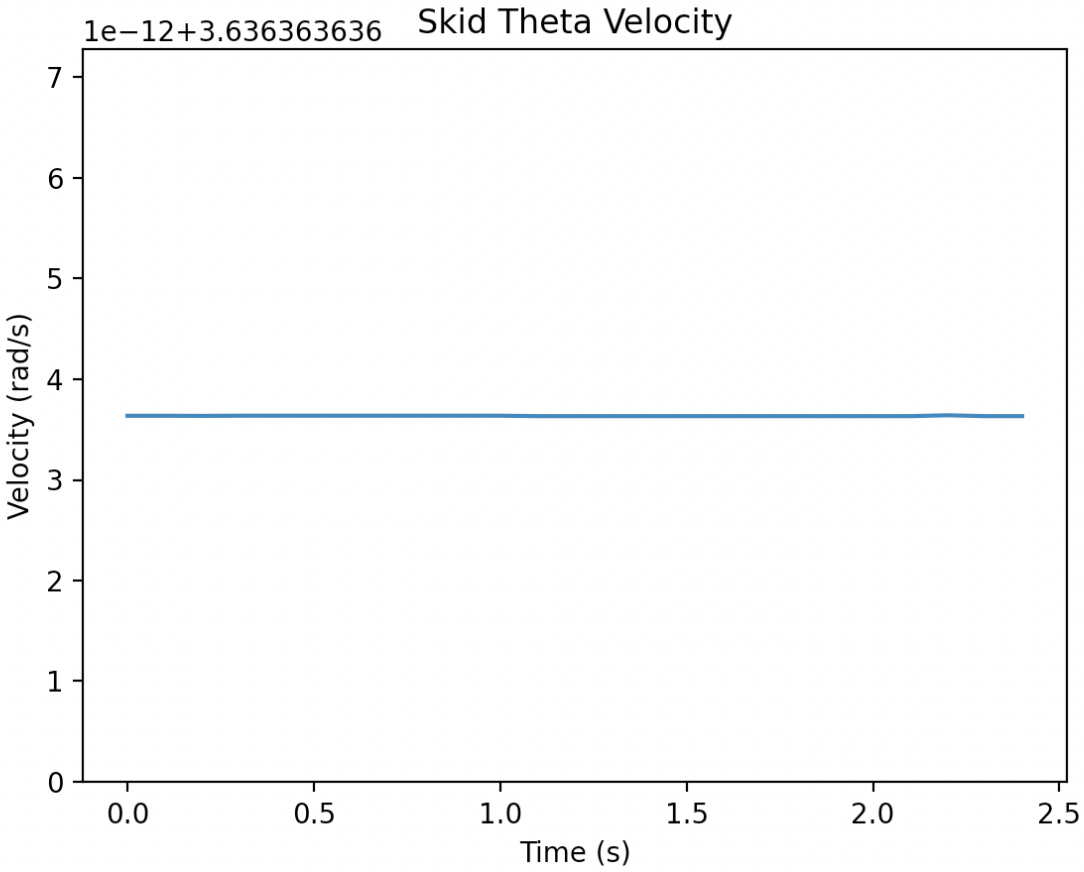
For 0.865 seconds {
    Move left track 3.54 m/s
    Move right track 5.54 m/s
}

For 1.73 seconds {
    Move left track 8 m/s
    Move right track 10 m/s
}


Resulting Path
2
1
0
−1
−2
Y
−2
−1
0
1
2
X


Skid X Velocity
Velocity (m/s)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0.0
0.5
1.0
1.5
2.0
2.5
Time (s)

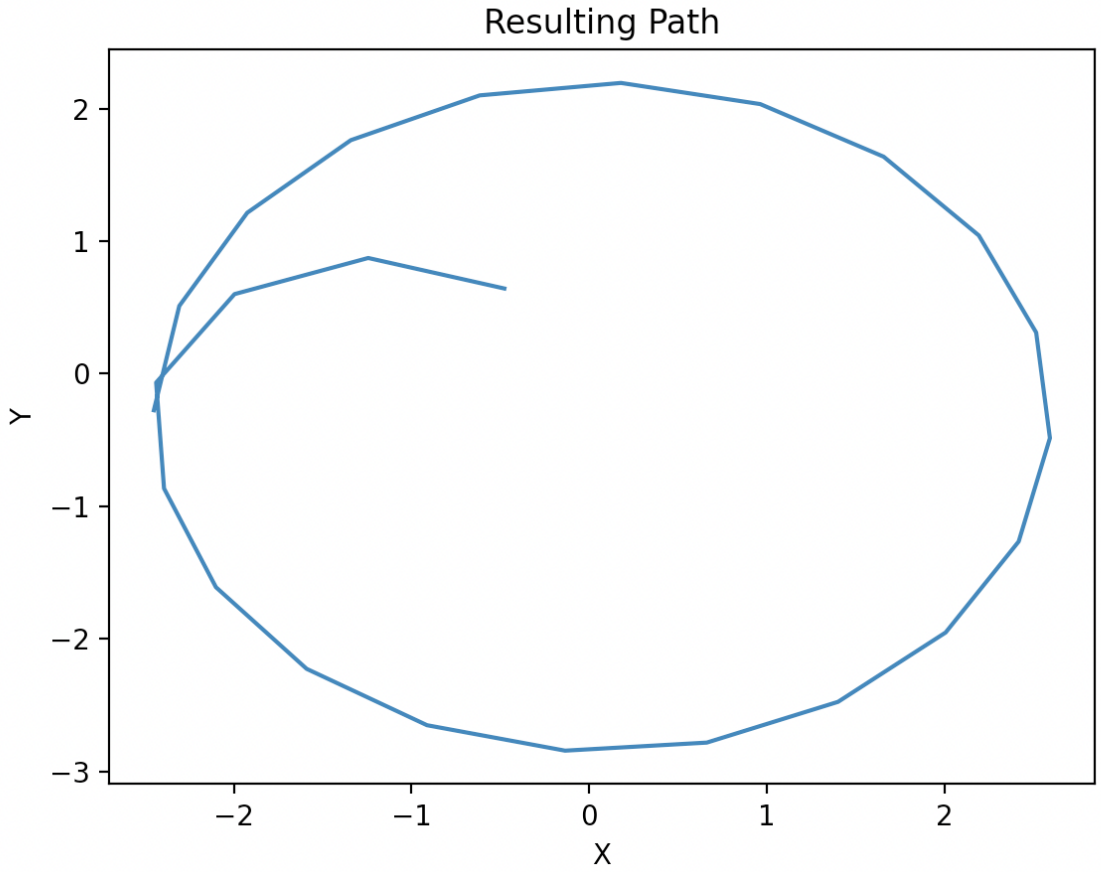

Skid Y Velocity
Velocity (m/s)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0.0
0.5
1.0
1.5
2.0
2.5
Time (s)

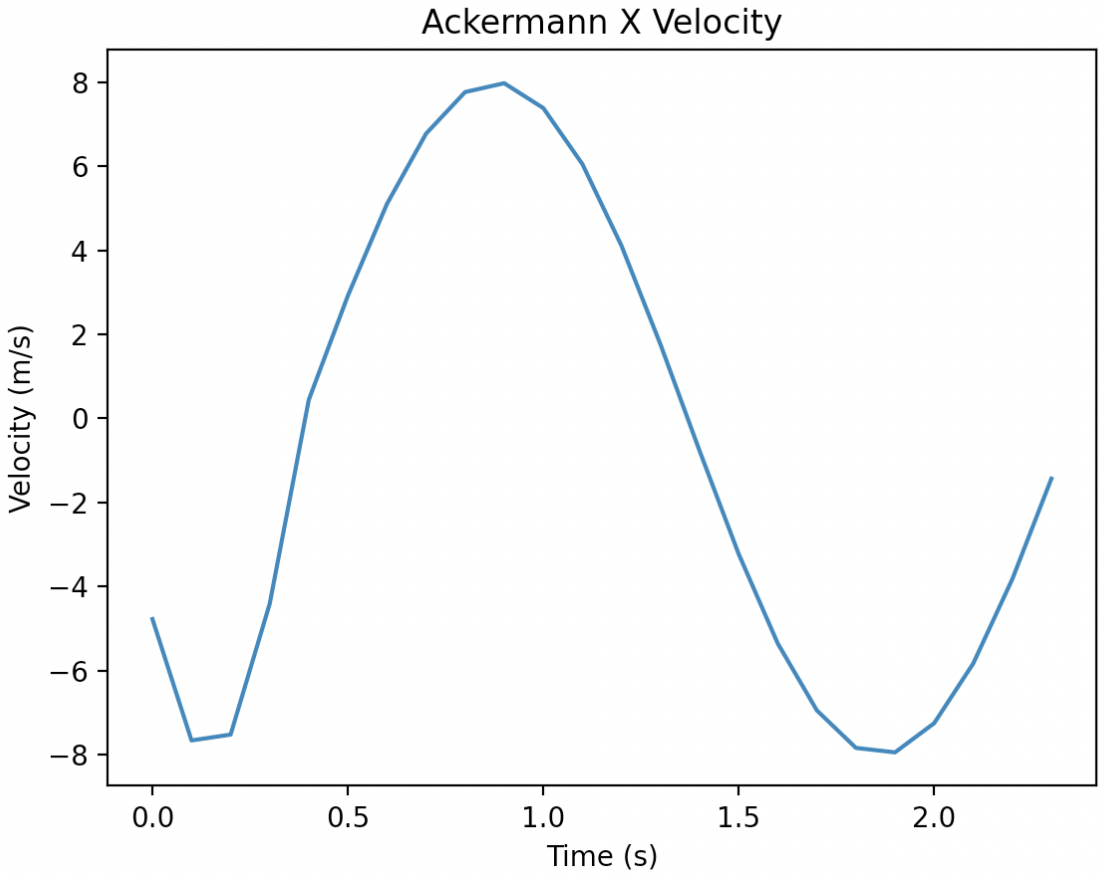


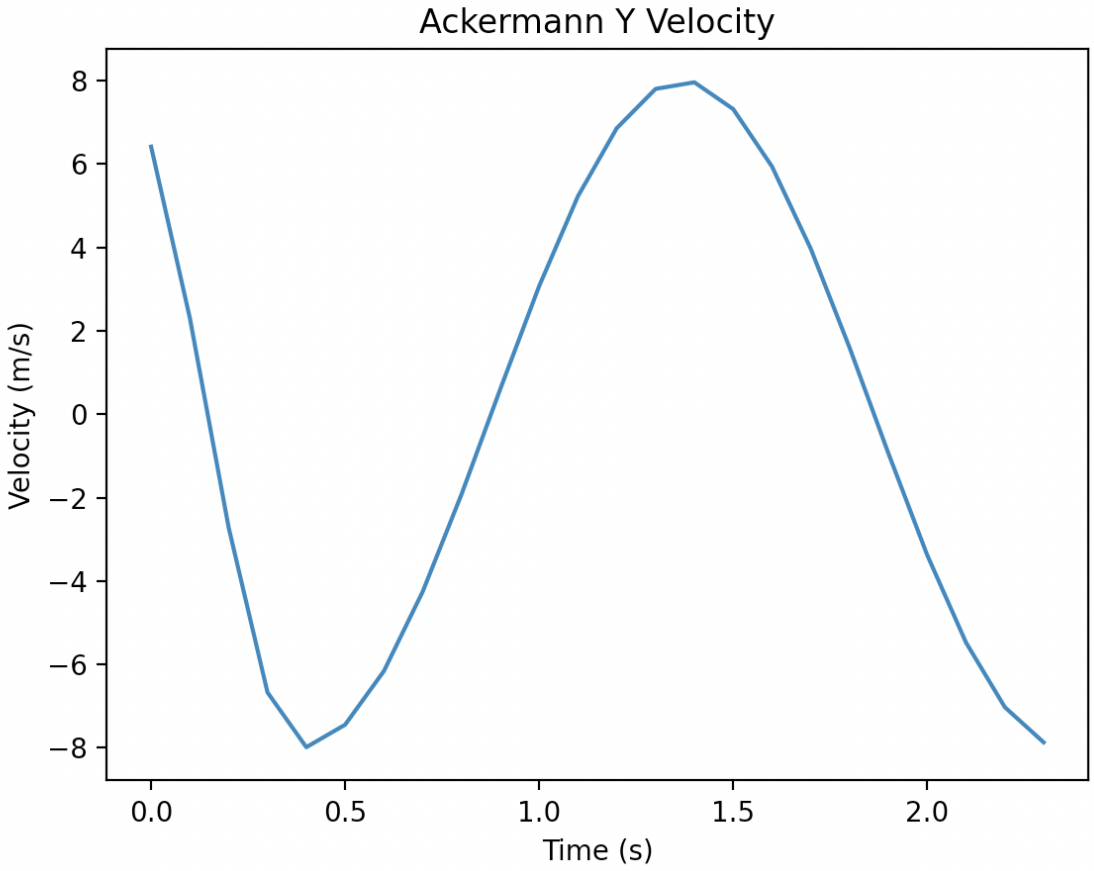
(b)

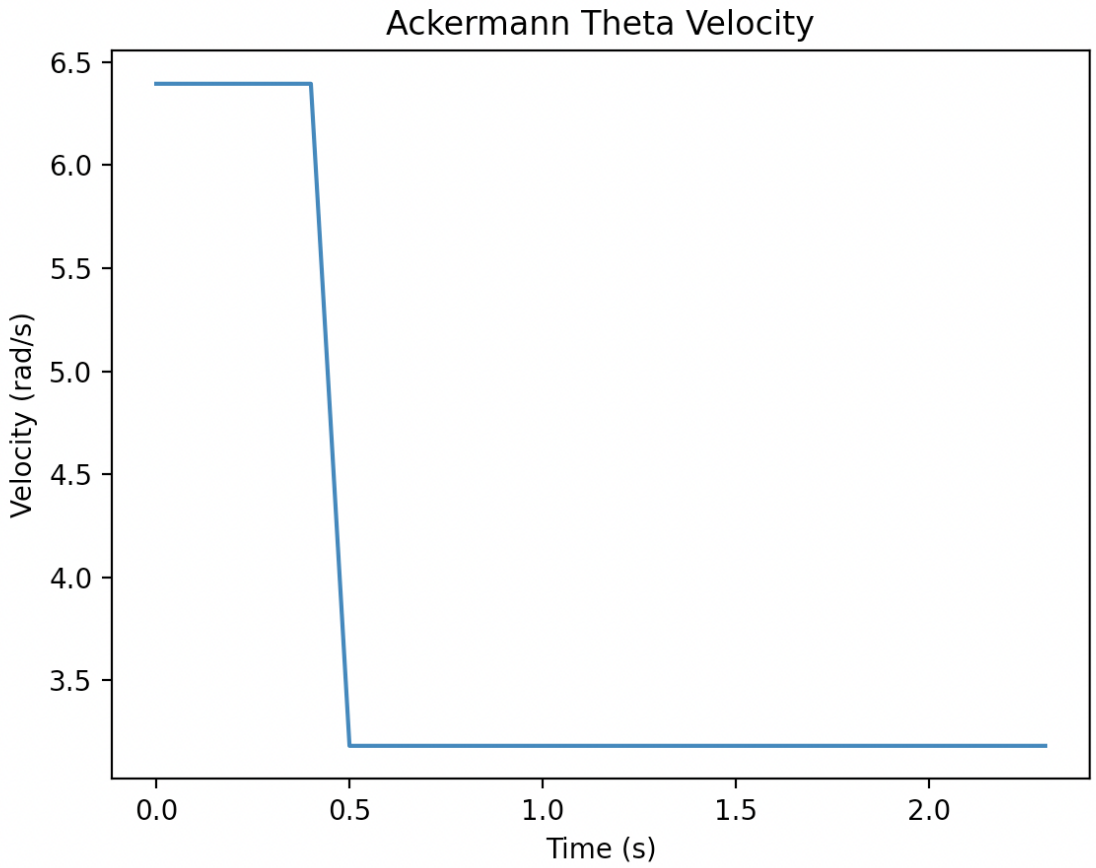
For 0.5 seconds {
    Set alpha to 0.54 radians
    Move at a velocity of 8 m/s
}

For 2 seconds {
    Set alpha to 0.29 radians
    Move at a velocity of 8 m/s
}

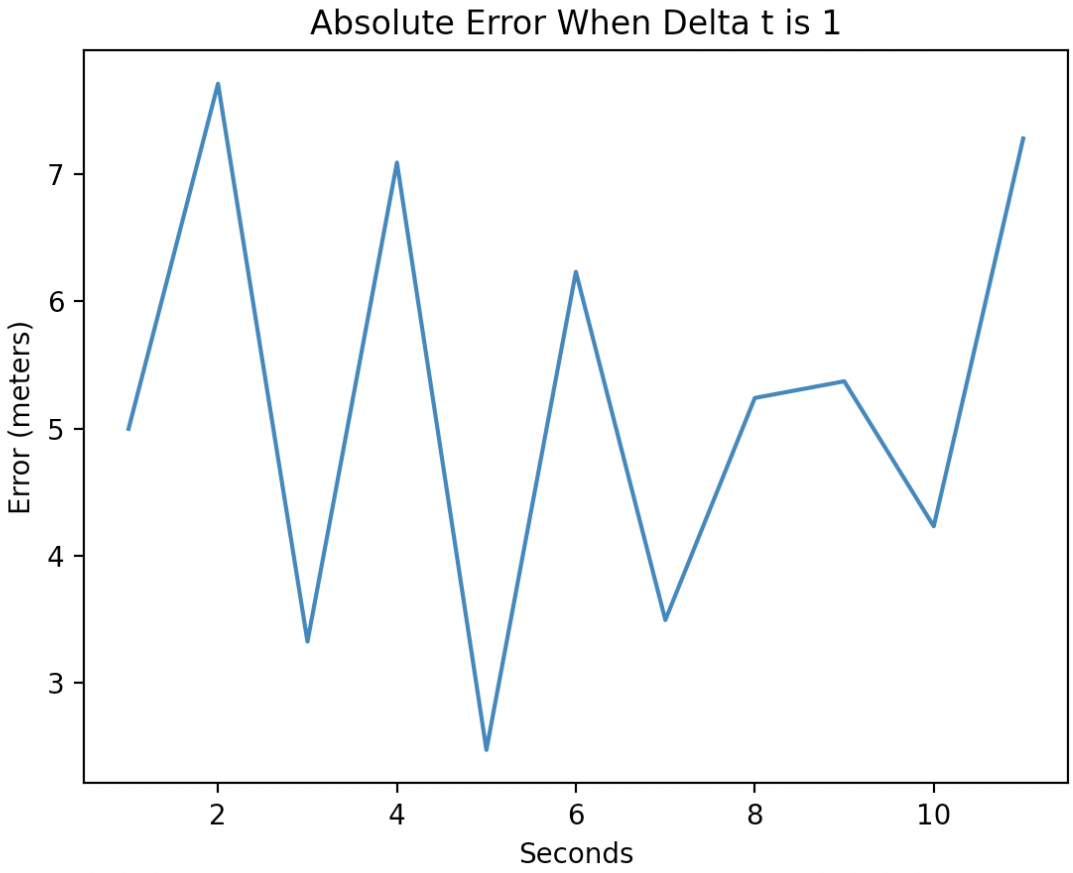

Resulting Path
2
1
0
−1
−2
−3
Y
−2
−1
0
1
2
X

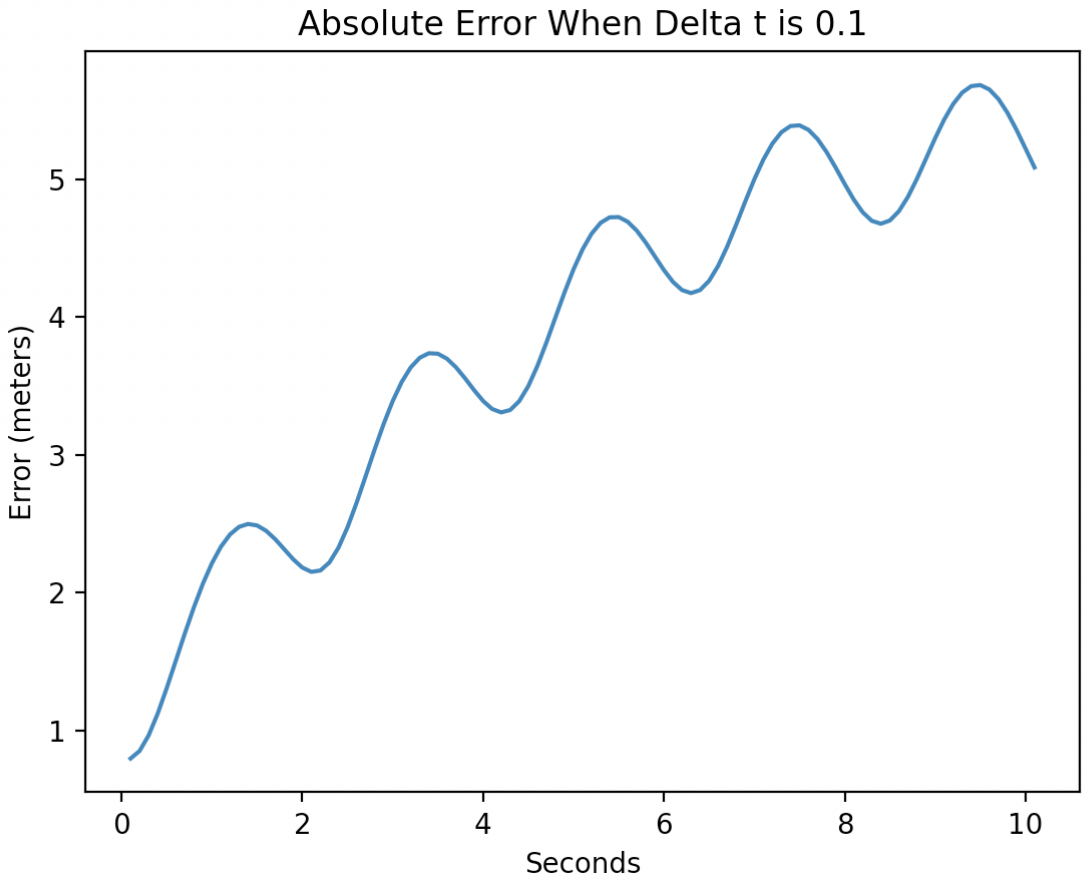

Ackermann X Velocity
Velocity (m/s)
8
6
4
2
0
−2
−4
−6
−8
0.0
0.5
1.0
1.5
2.0
Time (s)

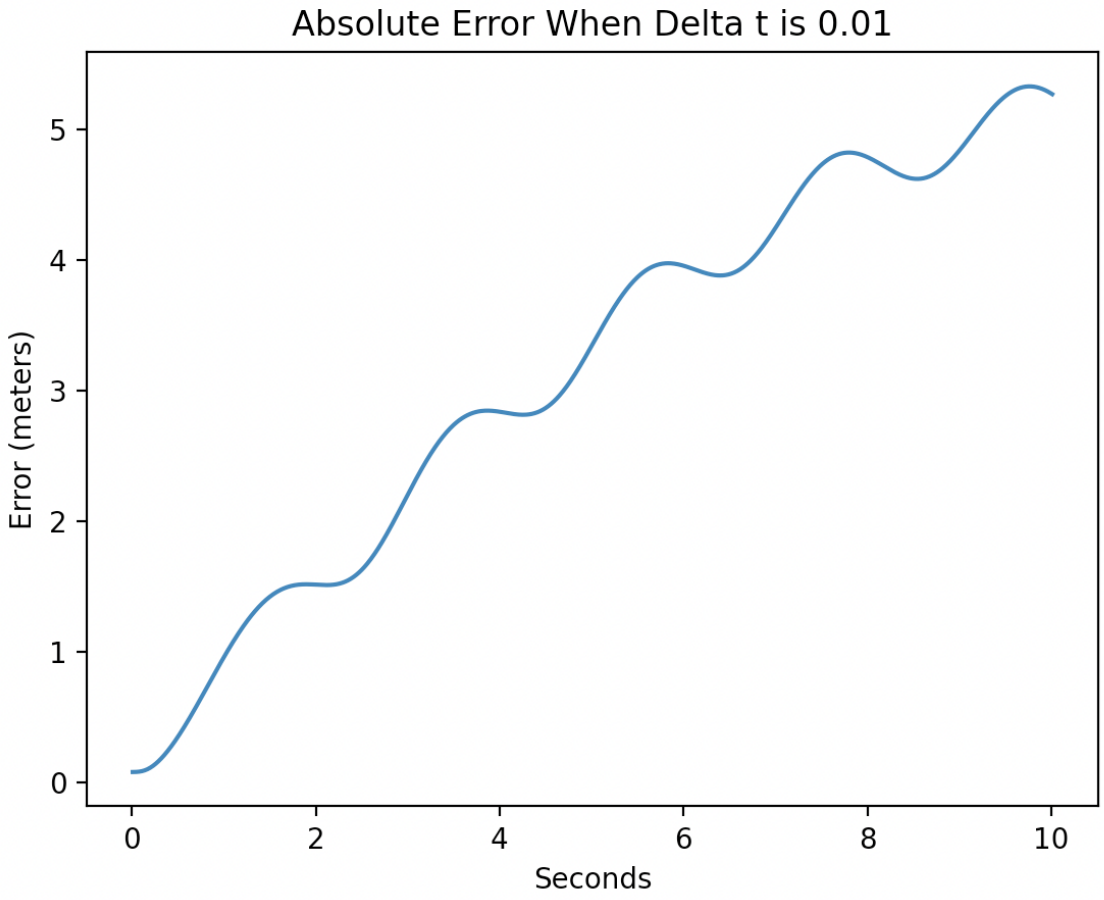

Ackermann Y Velocity
Velocity (m/s)
8
6
4
2
0
−2
−4
−6
−8
0.0
0.5
1.0
1.5
2.0
Time (s)



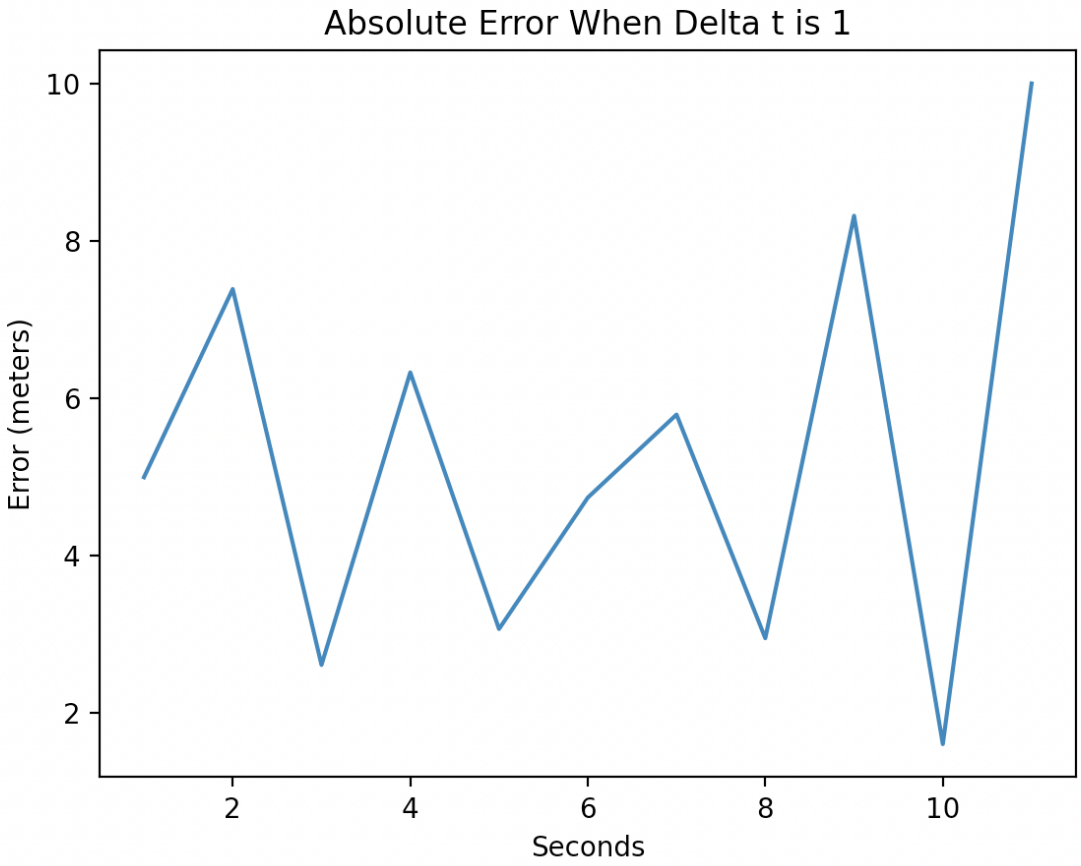
(c) Error calculations without slip

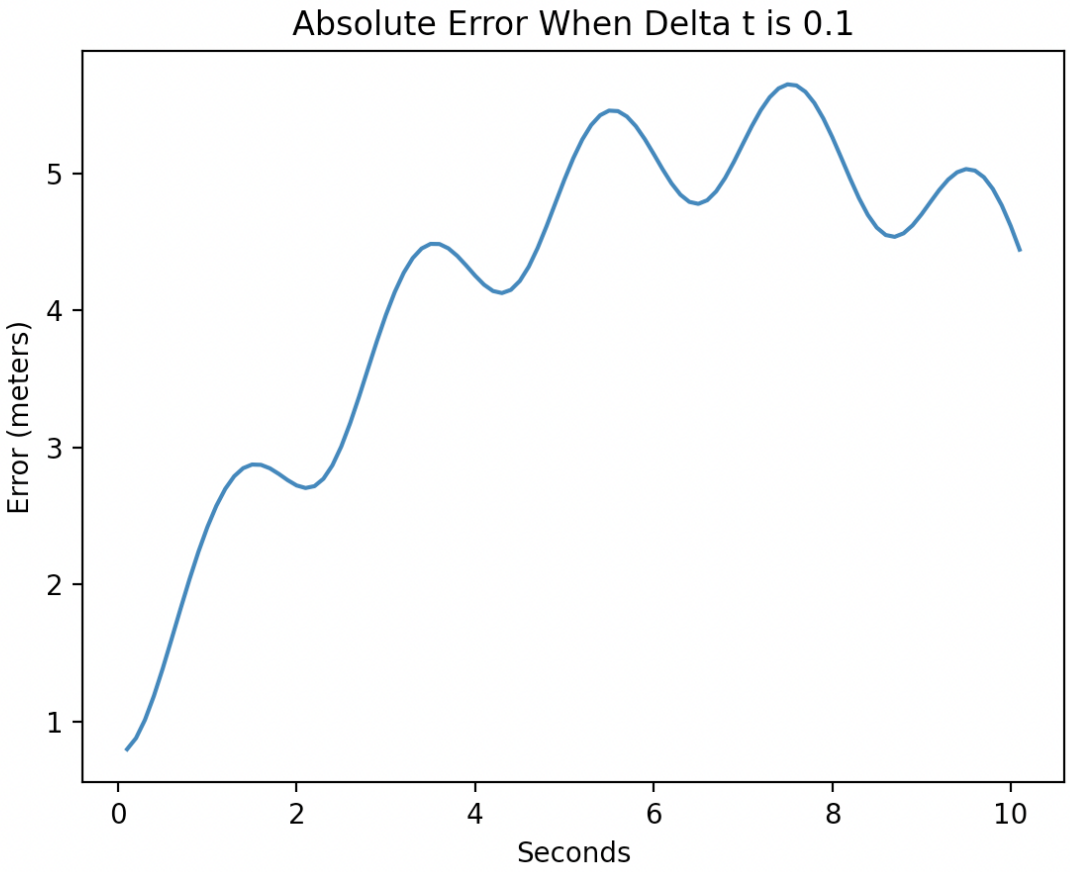

Absolute Error When Delta t is 1
Error (meters)
7
6
5
4
3
2
4
6
8
10
Seconds

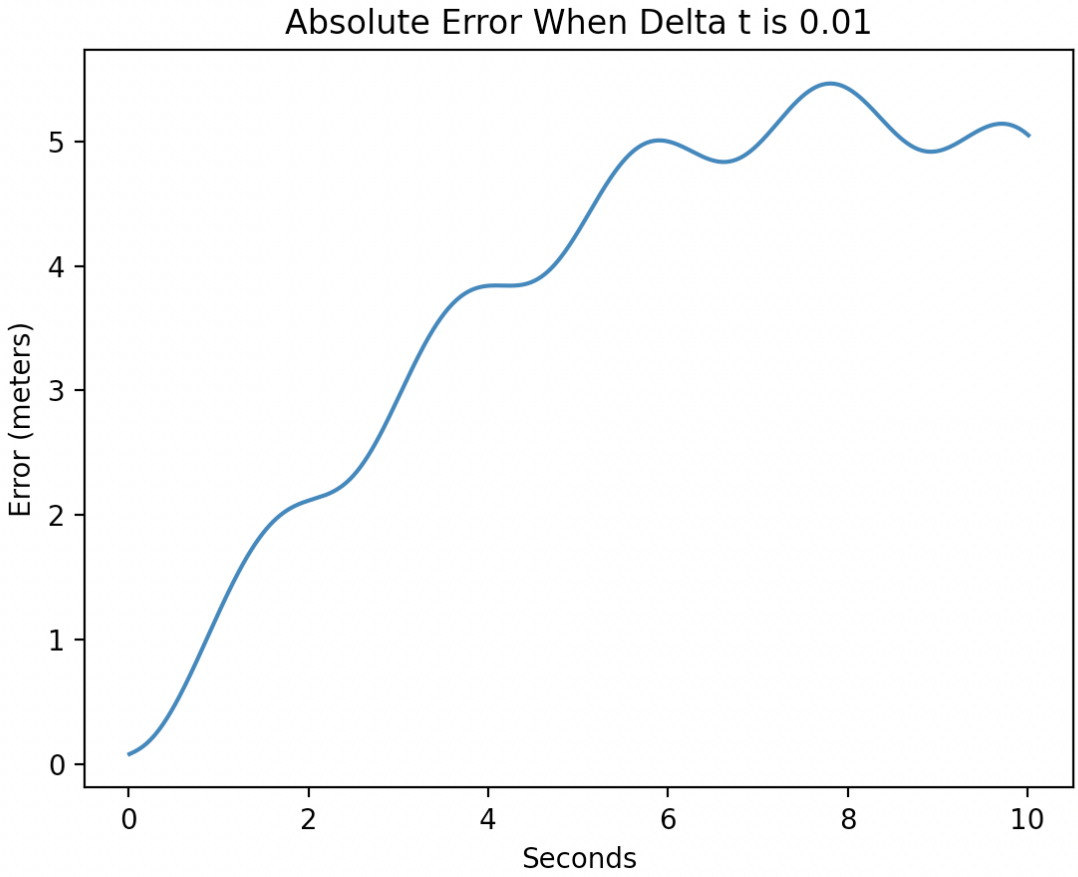

Absolute Error When Delta t is 0.1
Error (meters)
1
2
3
4
5
0
2
4
6
8
10
Seconds



(d) Error calculations with slip


Absolute Error When Delta t is 1
Error (meters)
10
8
6
4
2
2
4
6
8
10
Seconds


Absolute Error When Delta t is 0.1
Error (meters)
1
2
3
4
5
0
2
4
6
8
10
Seconds


Absolute Error When Delta t is 0.01
Error (meters)
0
1
2
3
4
5
0
2
4
6
8
10
Seconds

Problem 2

(a)

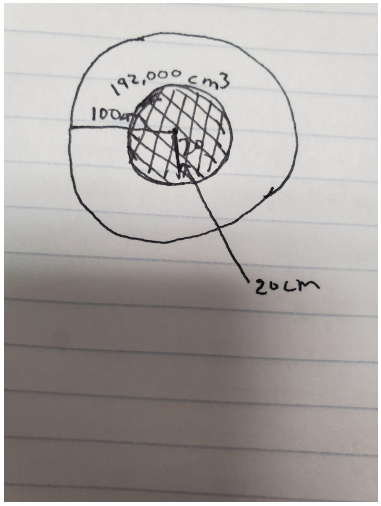
l_1 = 60cm , l_2 = 40cm, h = 20cm

r_total = 60cm + 40cm = 100cm

v_total = pi (100^2) 20 = 200,000 pi cm^3

v_unreachable = pi (20^2) 20 = 8,000 pi cm^3

v_actual = v_total - v_unreachable = 192,000 cm^3



(b)

DH Parameters:

cos(0+0) = 1	-sin(0+0) = 0	0	60(cos(0)) + 40(cos(0+0)) = 100
sin(0+0) = 0	cos(0+0) = 1	0	60(sin(0)) + 40(sin(0+0)) = 0
0	0	1	0
0	0	0	1

(c)

cos(30+45)	-sin(30+45) =	0	60(cos(30)) +

= .2588	-.9659		40(cos(30+45)) = 62.3143
sin(30+45) = .9659	cos(30+45) = .2588	0	60(sin(30)) + 40(sin(30+45)) = 68.6370
0	0	1	0
0	0	0	1

x = 62.3143, y = 68.6370

Problem 3

See midtermProblem3.py for the code.

(a)

Final Point (x, y, z): [1.200430338337864, 0.7992614577234782, 0.5001205311265852]
Final Thetas (degrees): theta1 = -48, theta4 = -76, theta5 = 119
Final Distances (meters): d2 = -0.669043, d3 = 1.409691

(b)

Final Point (x, y, z): [1.1968167477912097, 0.7962796822883315, 0.5068182904337003]
Final Thetas (degrees): θ1 = -54, θ4 = -239, θ5 = 86, θ6 = 40
Final Distances (meters): d2 = -0.321231, d3 = 1.386288, d6 = 0.200000

(c)

Final Point (x, y, z): [1.1970544163803107, 0.80821307455329, 0.49652305596327795]
Final Thetas (degrees): θ1 = -41, θ4 = -7, θ5 = 91, θ6 = 40
Final Distances (meters): d2 = -0.529513, d3 = 1.417562, d6 = 0.200000

Problem 4

(a)

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-F - m_p\, l\, \dot{\theta}^2 \sin\theta}{m_c + m_p}\right)}{l\left(\frac{4}{3} - \frac{m_p \cos^2\theta}{m_c + m_p}\right)}$$

theta = angle
theta_dot = angular velocity
theta_doubledot = angular acceleration
M_p = mass of pole
M_c = mass of cart
l = length of pole
g = gravity constant

This equation calculates the angular acceleration of the pole that is attached to the cart. It takes into account gravity and the circular motion that the pole will follow. If we were to code this the only changing variable would be force and theta. Everything else would be constants. The only variable the cart can change is the force applied.

$$\ddot{x} = \frac{F + m_p\, l\,(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta)}{m_c + m_p}.$$

F= force exerted
x = position
x_dot = velocity
x_doubledot = acceleration

In this equation we have a variety of variables including force applied to the cart, mass of the pole, length of the pole, angle of the pole, angular velocity of the pole, angular acceleration of the pole, and the mass of the entire cart including the pole. If we were to write the code for this equation the force applied to the cart, the angle of the pole, the angular velocity of the pole, and the angular acceleration of the pole can change as the simulation progresses. Everything else is constant throughout the simulation. The only variable the algorithm has control of is the force it applies in a specific direction. The angle, acceleration, and velocity of the pole are all results of that change in force.

(b)
The general idea for the controller is as follows:

```
# determine the direction we need to move based on the pole's angle
    if (poleAngle > 0.15):
        action = 1
    elif (poleAngle < -0.15):
        action = 0
    # Once the pole angle is close to center start moving the other
way to counteract its motion
    elif (poleAngle > -0.15 and poleAngle < 0 and action == 1):
        action = 0
    elif (poleAngle < 0.15 and poleAngle > 0 and action == 0):
        action = 1
    # Try to maintain angularVelocity as close to 0 as possible
    if (poleAngularVelocity > 0.25):
        action = 1
    elif (poleAngularVelocity < -0.25):
        action = 0
    # If we are moving too far away from center, periodically move
the cart back towards the middle
    if(i % 7 == 0 and abs(cartPosition) > .1):
        action = 1 if cartPosition > 0 else 0
```

The full script is included under Midterm/problem4/midtermProblem4.py.

(c)

The cart cannot recover once the angle of the pole passes ±8.28 °. See midtermProblem4.py for more details. We changed some of the values in the cartpole.py file so that the cart we were testing used the correct mass, force, etc. The cartpole environment file is located in the repository in the “problem4” folder. It’s labeled cartpole_angle_failure.py

Problem 5

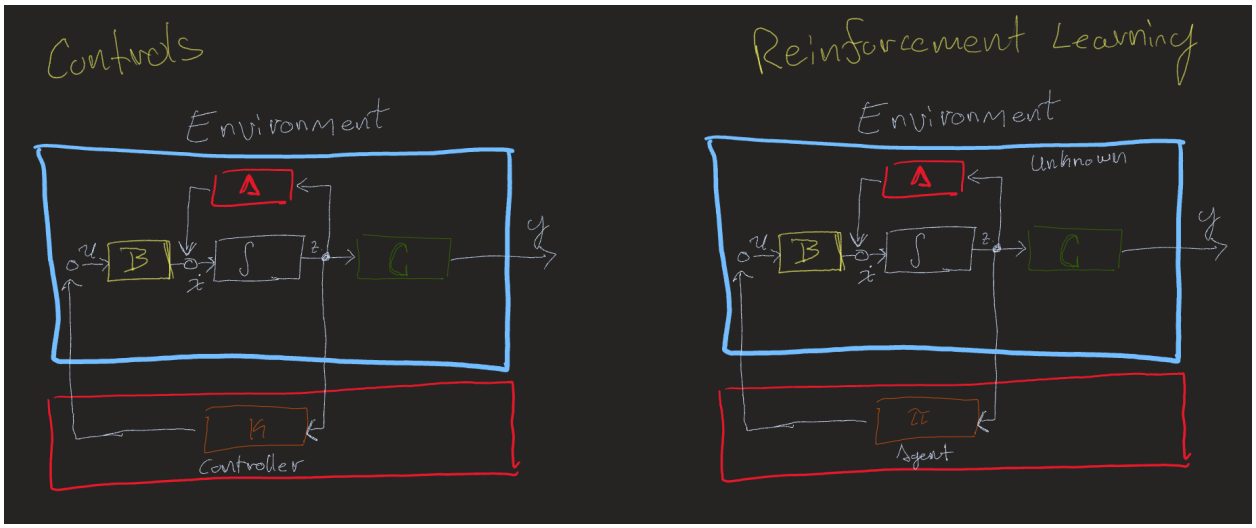
(a)
Merits:

1. Easier to solve your control problems because it will learn to solve the problem without you manually solving it.
2. It is broadly applicable because it can learn to solve the problem on various systems without explicitly reworking the problem. (This may often involve changing the control parameters, but the framework can be reutilized)
3. Reinforcement learning can find solutions that you have not thought about before.

Demerits:

1. Long training times.
2. Potential for incorrectly rewarding behavior.
3. It can be difficult to train an algorithm that will converge to useful behavior.

(b)



The biggest difference is with controls the environment is known, where as with reinforcement learning the environment is unknown. Reinforcement learning relies on reward whereas controls relies on feedback and corrective action from that feedback. Another major difference is that the algorithm in controls is known where the algorithm for the reinforcement learning is unknown.

(c) See Midtem/problem5/midtermProblem5.py for the code. We have included a screen recording of the Reinforcement Learning in the problem5 folder.

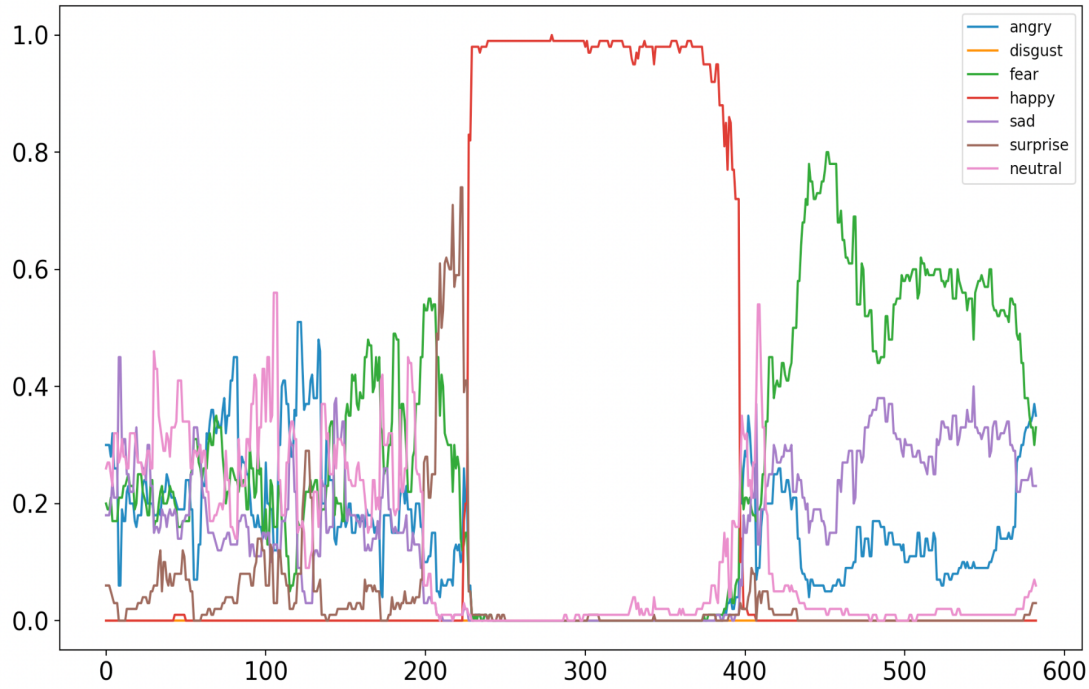
(d) See Midterm/problem5/problem5d for the code.

Mujoco.py is the script to begin training the model. OriginalHalfCheetahRL.mp4 displays the result of the training without penalizing hip-motor movement. HalfCheetahPenalized.mp4 displays the result of the training after penalizing hip-motor movement.

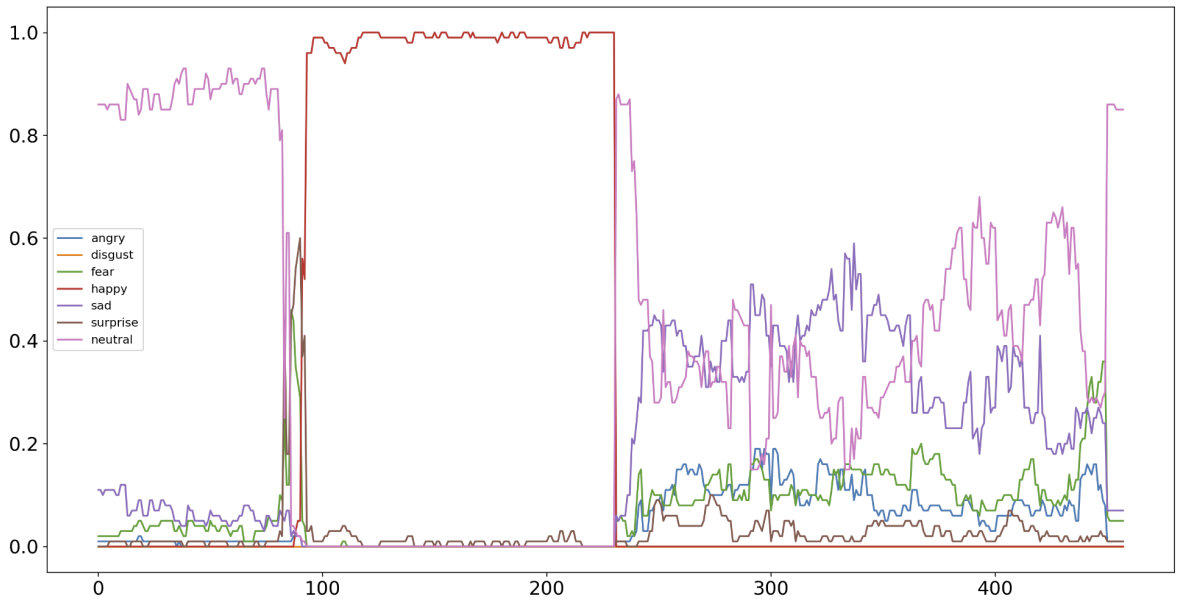
Problem 6

(a) See ./problem6/midtermProblem6.py for code implementation.

Video 1 graph:



Video 2 graph:



(b) See ./problem6/midtermProblem6.py for code implementation. The webcam recording is saved under ./problem6/screen_recording.avi.

(c) There are many logical applications of using hardware and software to recognize facial expressions of people. An example of an application could be a restaurant. A restaurant owner could analyze people's facial expressions to determine if they are enjoying the atmosphere and the food served there. Another example is for web applications. A website could track a user's facial expressions to figure out pain points that the website causes to the end user. By tracking this the developers of the website can make changes to alleviate the pain points.

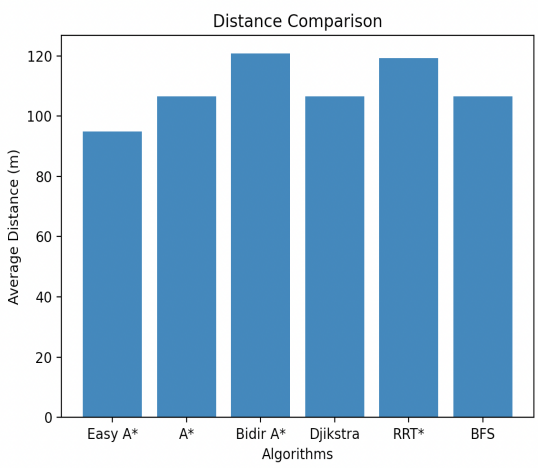
Using this technology does involve possibilities of invasion of privacy to the end users which could lead to law-suits against the company using this technology. Even if a developer has good motives when using the software, they also risk having hackers use their tech for malicious actions.

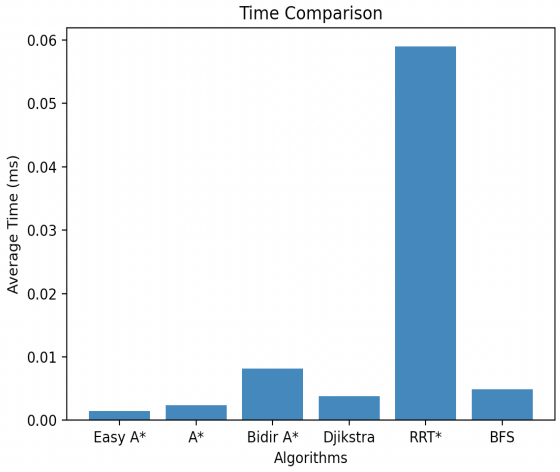
(d) See ./problem6/midtermProblem6.py for code implementation. The code is implemented to analyze a picture with two faces. It is included in the problem6 folder.

Problem 7

(a)

Algorithm	Average Time to Convergence (milliseconds)	Average Distance (or cost)
Easy A*	0.00014937	9.4853
A*	0.00025126	10.6569
BiDirectional A*	.0.00083632	12.0711
Djikstra	0.00038822	10.6569
RRT*	0.00344705	12.2140
BFS	0.00048908	10.6569





Which planner provided a path with the lowest cost on average?

The Easy A*, or the one we implemented separately from the github planners, was the path with the lowest cost on average. Second place went to the other A* implementation that we got from github.

Which one found a path the fastest on average?

The Easy A*, or the one we implemented separately from the github planners, was the fastest path on average. Second place was a tie between the A*, Djikstra, and BFS implementations we got from github.

After comparing your planner to these five other ones is there anything you would change in your planner to help it converge faster or find a path with a better cost?
After doing some research online we learned that by smoothing the path to reduce the right angle turns and expanding the distance around obstacles to avoid collision result in faster and more reliable a* planning algorithms. These would help us with our project as we plan to use some sort of planning algorithm with the ants in the robotics lab.

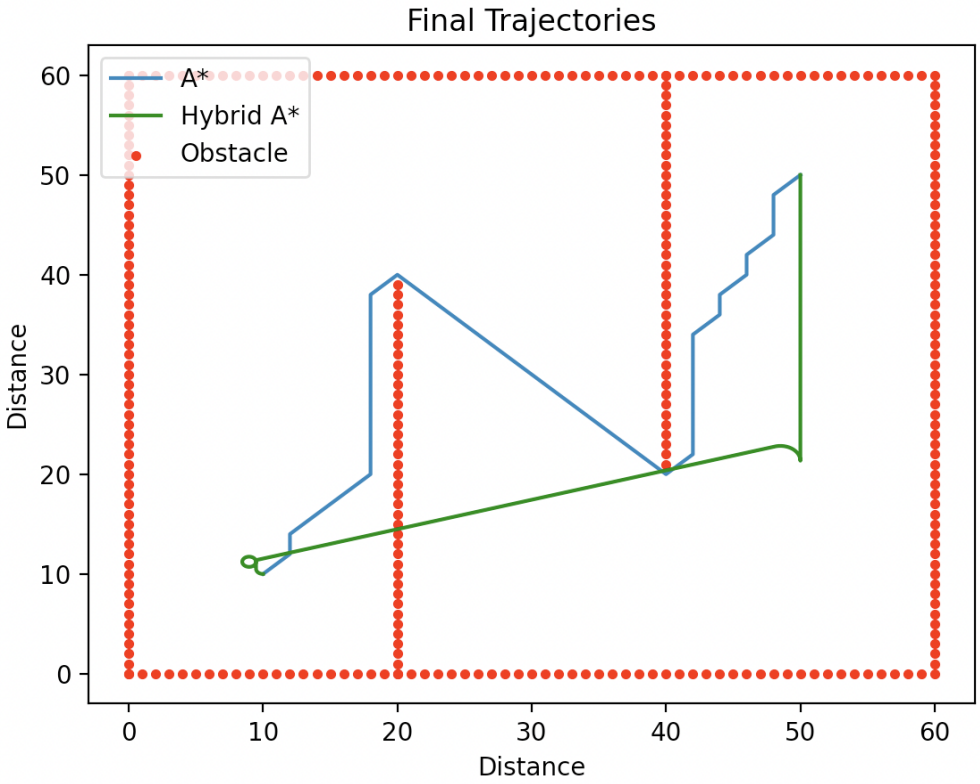
Which planner appears to be the best overall? Which planner would you use for a robot in a complex environment?
It seems that the A* is the best overall as the Easy A* and the A* from github were the first and second best out of all of the algorithms. We would start by using the A* for a robot in a complex environment since it performed the best here. That being said, the BiDirectional A* algorithm seems to perform well with multiple obstacles so we would likely also use this as we are testing.

(b)

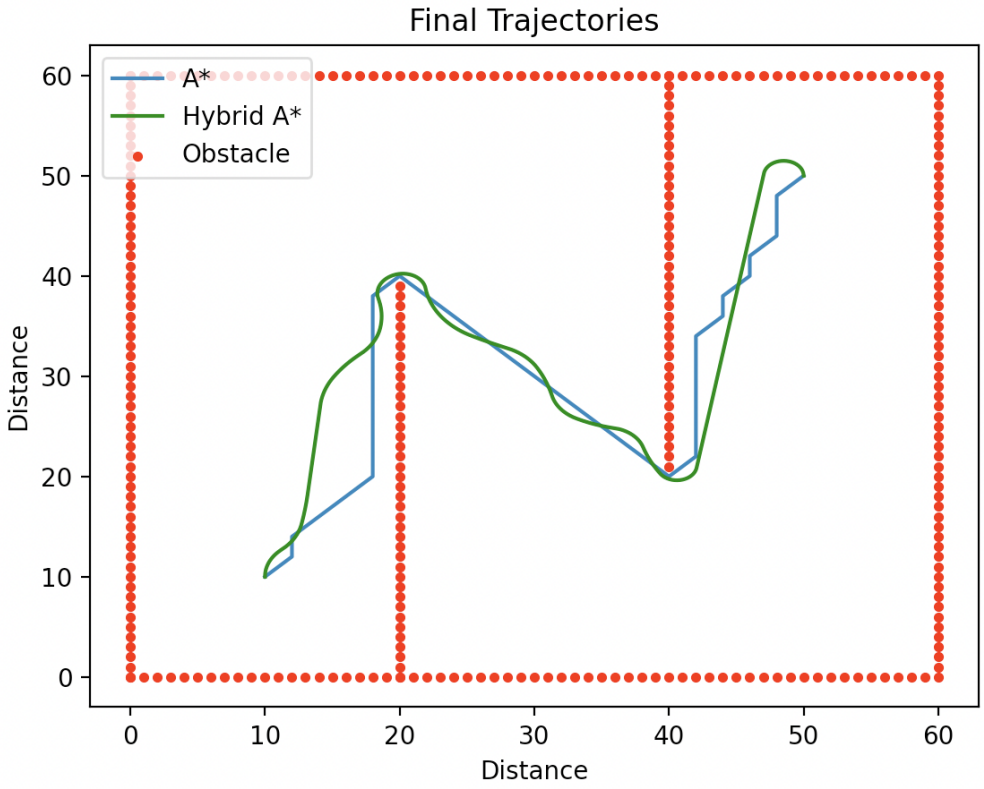
The main algorithmic difference is the introduction of the skid steer kinematics into the path planning as it changes what positions are available to traverse next. As implemented, the Hybrid A* algorithm has limited options for next traversal node since the car can only turn so far. This results in additional functions that get the available nodes that can be accessed. The A*, however, checks the area surrounding the current location and is not restricted by turning limits.

The Hybrid A* also takes advantage of the Reeds Shepp path planner which assists with the vehicle path planning that takes place. It also uses a much larger heuristic calculation function than the A*. To give some context, the A* uses only 3 lines of code for while the Hybrid A* uses over 60 lines of code.

The code for this problem is included in Midterm/problem7. We tried getting the skid steer implemented but we were running into a variety of problems. Changing the calculations of the move function (line 100 in car.py) resulted in the car performing multiple circles before cutting through the obstacles. We assume there is some sort of error with the obstacle detection if this is the case, but that doesn't explain the circles at the beginning. Here is a graph of the skid steer attempt:



Here is the graph of the two path planning algorithm's final trajectories with ackerman steering:

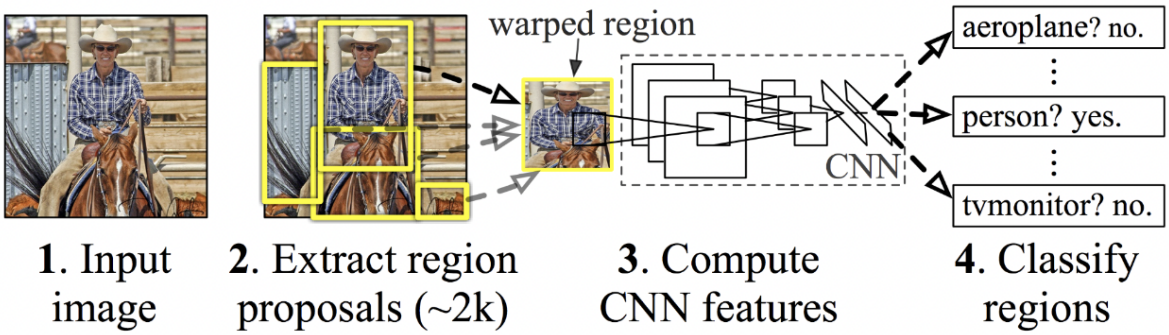

Final Trajectories
A*
Hybrid A*
Obstacle
Distance
Distance
0
10
20
30
40
50
60

Problem 8

a) Problem8a.ipynb, the pictures are in problem8/agri_data/data/
b) In folder problem8b. Read the readme.md
c) In folder problem8c. Read the readme.md
d) We tried to implement Detectron but could not get the required dependencies to work with our machines. Mostly M1 Macs. Below is a comparison of R-CNN, Faster R-CNN, and RetinaNet.

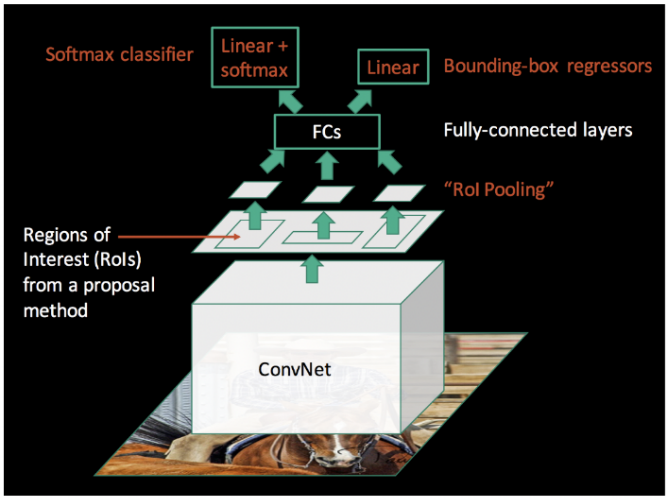
R-CNN

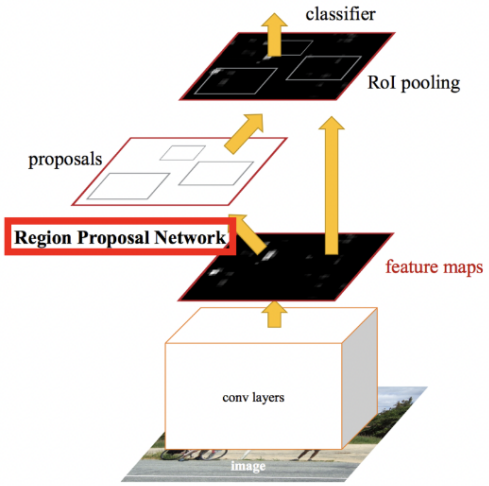
- Huge time to train 2,000 region proposals per image
- Can not do real-time detection because it take too long to process
- Least accurate among the three



Faster R-CNN

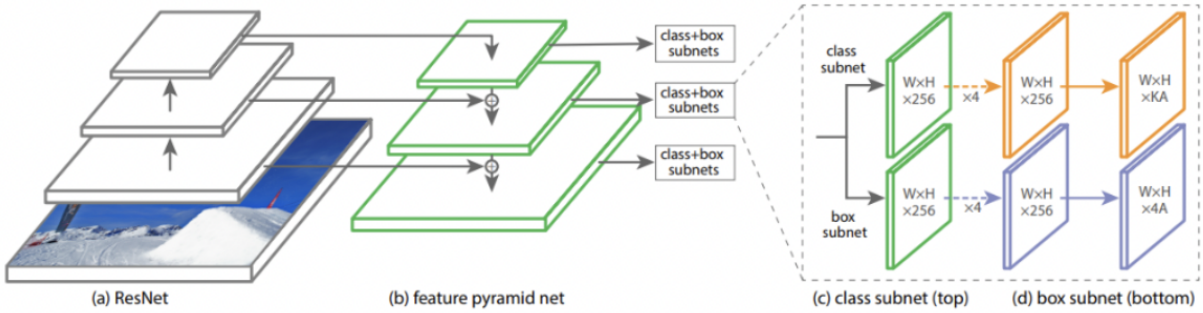
- Can do real-time
- .2 second prediction time
- Second accurate among the three





RetinaNet

- Most accurate among the three
- Can not do real-time detection



Problem 9

a. A merit of this is that there would be multiple advances in certain areas of robotics where ethical restrictions currently exist. Military, privacy, and security are just three of those areas that would change. A demerit of this is that we would lose some of our personal privacy. This would be most noticeable with anything internet related. Our personal data could be collected, stored, and shared by a variety of robots using the IoT.

b. To implement and respect Issac Asimov's three laws of robotics, There are some additional complexities and precautions that need to be met. For every action a robot takes, it should confirm that its next action will not injure a human being, follow the exact instructions given from a human without hurting a human, and also ensure the action to not harm the robot.

   An example of this would be a stationary archery robot that can pivot to shoot at a field target. Pseudo code would look like the following:

```
shootArrow() {
    if(humanIsNotInTheWay() && pivotAngleNotDangerousToRobot()) {
        shoot();
    }
    else {
        wait(10)
        shootArrow()
    }
}

acquireFieldTarget() {
    if(!targetIsHuman) {
        setTarget()
        shootArrow()
    } else {
        return
    }
}

processCommand(command) {
    if(command.targetHuman) {
        return
    }
    else if(command.harmRobot) {
        return
    }
    else if( command.acquireFieldTarget) {
        acquireFieldTarget()
    }
    else {
        return
    }

}
```

c. The first article by Callahan-Law basically states that liability is indeterminate and responsibility could go to anyone. The lack of regulation has left a lot of these legal questions up in the air. We agree that it is an extremely difficult question to answer, especially when these technology companies are doing their best to create breakthrough technologies. The person or entity responsible for an autonomous vehicle could depend on what the manufacturer sells the vehicle as and the state's policies. The scenario addressed in this article is when an autonomous vehicle strikes a pedestrian. If the

manufacturer guarantees that the driver does not need to be attentive during the driver's travel, then the manufacturer should be at fault for the crash. If the manufacturer says the driver should still pay attention to his surroundings, then the driver should be at fault.

The wikipedia article outlines tort liability and three basic theories for its implementation: traditional negligence, no-fault, and strict liability. We think a combination of all three would be ideal, and should be entirely based on the situation at hand. The article also states that research from the Institute for Highway Safety has shown a decrease in collisions thanks to some of these advances. There are bound to be some growing pains when developing new technologies, and that needs to be taken into account when establishing liability.

d. A law that may be helpful for regulating autonomous vehicles could involve an extremely steep set of requirements of each product. This would involve extensive testing and proof of accuracy to back the requirements up. Hopefully this would result in autonomous vehicles with a low margin of error for their object detection and movement controls. The goal of this law is to help prove that the vehicle is safe for the public before it ever gets on the road.

   Another law could state that if a driver chooses to ride in an autonomous vehicle they also take responsibility for the car's actions. This would be important because a driver also has the capability to add inputs to the vehicle's controls. At the point of a collision, it would be hard to tell if the user was manually steering the car or have the car take control over all the cars movements.

   The drawbacks to having these added laws is that the development of autonomous vehicles will be slowed to meet the new strict required safety regulations. This means that autonomous vehicles would not be seen on the road for a couple years. Additionally, this puts more strain on the manufacturers which will likely drive up the price of these vehicles. The other drawback is that there would inevitably be drivers at fault for a collision that they had no control over.