Robot Intelligence **Name: The Black Spot**
Fall 2022
Midterm Exam
8/31/2022
Take Home Exam - Due Before Class on Canvas 10/31/2022

This exam contains 25 pages (including this cover page) and 9 questions.
The total number of points is 240. Graduate students must answer some additional questions. Undergraduates answering graduate questions will be given additional points and a feeling of satisfaction from attempting difficult things (probably).

Any programming/code artifacts associated with this exam should be submitted along with the final PDF to canvas. Please put your entire submission in a single .zip file.

This exam will cover topics in motion, planning, sensor processing, and ethics.

Grade Table (Quick view to see the breakdown)

Question	Points	Score
1	30	
2	20	
3	30	
4	20	
5	30	
6	30	
7	30	
8	30	
9	20	
Total:	240	

---

1. (30 points) Moving in a car

   I would like you to use a skid steer model of a robot that is $75cm$ long and $55cm$ wide and run a few simple experiments with it. Please upload your resulting figures and python (or other) code:

In order to generate code that can take in the above parameters while utilizing the equations of motion given in class, some bridge equations are needed. This is because the equations of discretized robotic motion either take the individual velocities of the wheels or the average velocity and the steering angle, which do not involve the car's driving radius. Yet, the driving radius which the car must match given a series of commands (in the form of the discretized motion equations) is the requirement. Therefore, it is necessary for some bridge equations to relate driving radius to wheel velocity. Below are the primary bridge equations.

$$\theta = \frac{v_r - v_l}{w} \cdot dt, \tag{1}$$

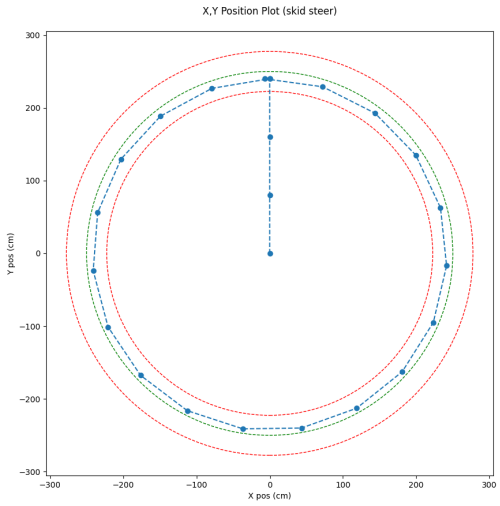
$$\theta = \frac{\ell}{r} = \frac{v_{\text{avg}}}{r} \cdot dt \tag{2}$$

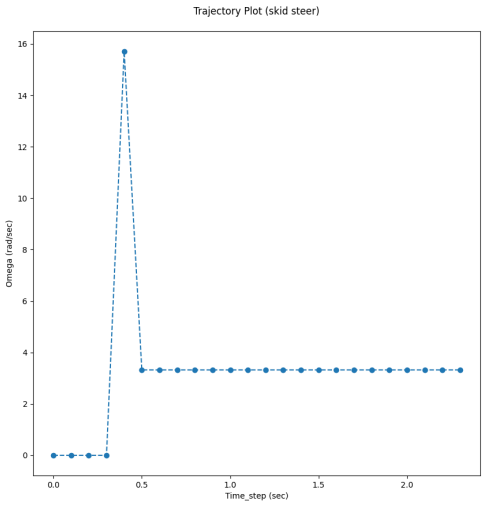
$$r = \frac{\ell}{r} = \frac{v_{\text{avg}}}{\theta} \cdot dt = \frac{w}{2} \cdot \frac{v_r + v_l}{v_r - v_l} \tag{3}$$

Equation 1 is the discretized motion equation for change in heading. Equation 2 is the definition of a radian (arc length / radius), which is substituted using $d = rt$ to include a average velocity of the robot. Equation 3 shows the reordering of equation 2, then the substitution of equation 1 and the decomposition of v_{avg} to get a formulate relating radius to wheel velocity. These equations are used to solve the following problems.

   (a) (5 points) Make a list of commands (at t=0.1) that will allow this robot to traverse along the edge of a $5m$ diameter circle. The robot starts off in the center of the circle (0,0), and you cannot leave the circle's border. Plot both the resulting path (x, y) and trajectory (x, y, and angular velocities). Assume a constant velocity of $8m/s$

The link to the responsible code, as well as all figures, can be found here:
Link: `https://github.com/ColtonChill/cs6510-midterm/blob/dev/problem01/part_a.py`





Commands

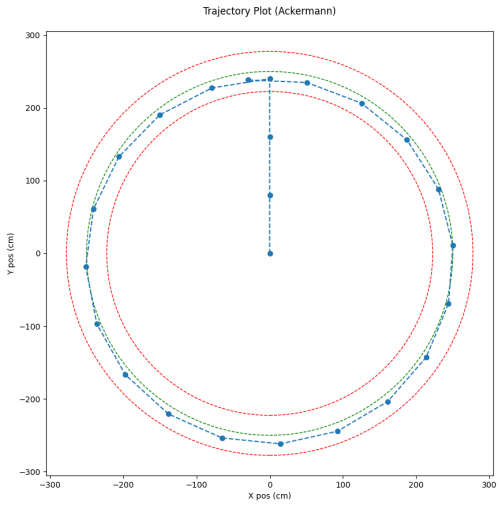
time(sec)	left_vel(m/s)	right_vel(m/s)	x-pos(cm)	y-pos(cm)	omega(rad/sec)
0.0	0	0	0.0	0.0	0.0
0.1	800	800	0.0	80.0	0.0
0.2	800	800	0.0	160.0	0.0
0.3	800	800	0.0	240.0	0.0
0.4	-431.9689898685966	431.9689898685966	0.0	240.0	15.707963267948967
0.5	708.6505190311418	891.3494809688582	-78.8990992639406	226.77380839320747	3.3217993079584796
0.6	708.6505190311418	891.3494809688582	-149.17195552719537	188.54128736050527	3.3217993079584796
0.7	708.6505190311418	891.3494809688582	-203.1354555557243	129.48249745865442	3.3217993079584796
0.8	708.6505190311418	891.3494809688582	-234.88963005424515	56.05448916537789	3.3217993079584796
0.9	708.6505190311418	891.3494809688582	-240.96271287319536	-23.714663174822974	3.3217993079584796
1.0	708.6505190311418	891.3494809688582	-220.690718137102	-101.10359151357054	3.3217993079584796
1.1	708.6505190311418	891.3494809688582	-176.29003554024473	-167.6511638579704	3.3217993079584796
1.2	708.6505190311418	891.3494809688582	-112.61510672034962	-216.08156174794536	3.3217993079584796
1.3	708.6505190311418	891.3494809688582	-36.62767658005973	-241.09976433475333	3.3217993079584796
1.4	708.6505190311418	891.3494809688582	43.36435231601578	-239.97046672587 31	3.3217993079584796
1.5	708.6505190311418	891.3494809688582	118.6152442483368	-212.81713795406054	3.3217993079584796
1.6	708.6505190311418	891.3494809688582	180.89762428239857	-162.60852178818016	3.3217993079584796
1.7	708.6505190311418	891.3494809688582	223.40199848137974	-94.8340562871304	3.3217993079584796
1.8	708.6505190311418	891.3494809688582	241.48125351663003	-16.90369931778001	3.3217993079584796
1.9	708.6505190311418	891.3494809688582	233.1587376051339	62.66222133235068	3.3217993079584796
2.0	708.6505190311418	891.3494809688582	199.34437296899387	135.16455746198284	3.3217993079584796
2.1	708.6505190311418	891.3494809688582	143.73517168195116	192.676440862	3.3217993079584796
2.2	708.6505190311418	891.3494809688582	72.41103176313257	228.9099483494182	3.3217993079584796
2.3	708.6505190311418	891.3494809688582	-6.829993821910833	239.90357669635162	3.3217993079584796

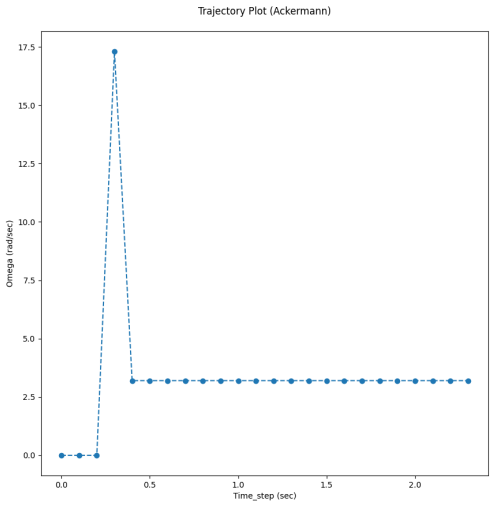
Above are the plots for the skid-steer commands. Each blue point on the graph represents the robot at a given dt. The first plot shows the (x, y) position at each time point, while the second shows the angular velocity experienced by the robot at those same time points.

Because of the given requirement that the vehicle must drive at a constant velocity and that the vehicle can not leave the circle (colored green), the robot will not travel to the edge of the circle in a clean multiple of $dt = 0.1$. Therefore, we only command the robot to go forward until $dt = 0.3$. This means that the vehicle is not technically tracing a circle focused on the origin, and this seemed to our team to be the best compromise.

(b) (5 points) Do the same as the above for a traditional car (Ackermann steering).

The link to the responsible code, as well as all figures are here:
Link: `https://github.com/ColtonChill/cs6510-midterm/blob/dev/problem01/part_b.py`





Commands

time(sec)	avg_vel(m/s)	alpha(rad)	x-pos(cm)	y-pos(cm)	omega(rad/sec)
0.0	0	0	0.0	0.0	0.0
0.1	800	0	0.0	80.0	0.0
0.2	800	0	0.0	160.0	0.0
0.3	800	1.0183444582138714	0.0	240.0	17.30242760614404
0.4	800	0.2914567944778671	-78.98522602251951	227.2982650723806	3.1999999999999997
0.5	800	0.2914567944778671	-149.96525899830306	190.3952175185988	3.1999999999999997
0.6	800	0.2914567944778671	-205.73355553033022	133.0375928997577	3.1999999999999997
0.7	800	0.2914567944778671	-240.6280071031954	61.048862882976806	3.1999999999999997
0.8	800	0.2914567944778671	-251.1058092262205	-18.26201696758804	3.1999999999999997
0.9	800	0.2914567944778671	-236.103159411025	-96.84267933768146	3.1999999999999997
1.0	800	0.2914567944778671	-197.14326414865678	-166.71489528325813	3.1999999999999997
1.1	800	0.2914567944778671	-138.18168902820787	-220.78459714405685	3.1999999999999997
1.2	800	0.2914567944778671	-65.20475347005896	-253.56213333013366	3.1999999999999997
1.3	800	0.2914567944778671	14.378255279321806	-261.71962791647337	3.1999999999999997
1.4	800	0.2914567944778671	92.48734088612689	-244.42885738220284	3.1999999999999997
1.5	800	0.2914567944778671	161.19215320077166	-203.4453391729344	3.1999999999999997
1.6	800	0.2914567944778671	213.5171500775015	-142.93009562347646	3.1999999999999997
1.7	800	0.2914567944778671	244.14981833574683	-69.02718881068398	3.1999999999999997
1.8	800	0.2914567944778671	249.98004878120946	10.760080931755551	3.1999999999999997
1.9	800	0.2914567944778671	230.4159029917955	88.33097882220511	3.1999999999999997
2.0	800	0.2914567944778671	187.44371233065525	155.80979645990809	3.1999999999999997
2.1	800	0.2914567944778671	125.4264073837696	206.34546591352603	3.1999999999999997
2.2	800	0.2914567944778671	50.66055326571137	234.80714291958517	3.1999999999999997
2.3	800	0.2914567944778671	-29.262935371497292	238.3051372103151	3.1999999999999997

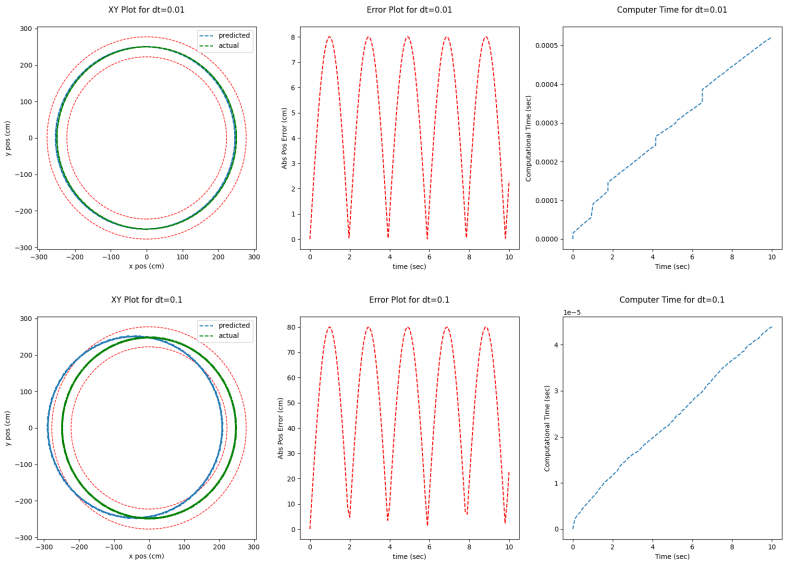
Like part A, the above plots are generated by these Ackermann commands, with the same temporal relation between plots.

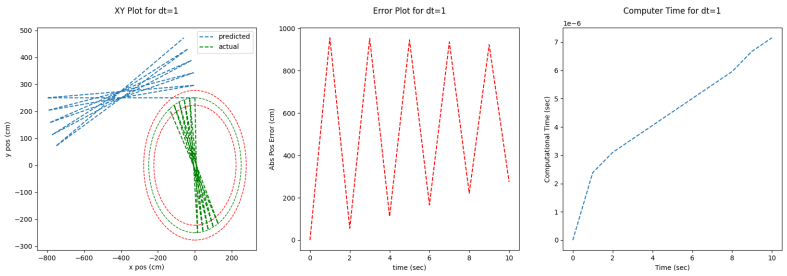
(c) (10 points) Your Ackermann vehicle (with the same dimensions as described for the skid steer), is driving on a circle of radius 2.5m for 10 sec. Assume that you begin on the edge of the circle. Calculate the positional error with our computational approximation using the forward Euler method, as referred to in the course notes and illustrated in Reading 2, eq. 1.6. Graph the errors and computing time for three different time-steps ($\Delta t = 1, 0.1, 0.01$), error is defined as the absolute distance between the expected (from equations) to real x, y position (defined analytically) over time.

Brief notes on what this problem is about: Imagine that you are moving a semi-truck and you have GPS positions (somewhat accurate) and an internal prediction model. We want the internal models to be updated based on "ground truth" information to build better estimates of the vehicle motion over time.

The link to the responsible code, as well as all figures are found below:
Link: `https://github.com/ColtonChill/cs6510-midterm/blob/dev/problem01/part_c.py`

Here, we can see a clear increase in the magnitude of error as dt increases.



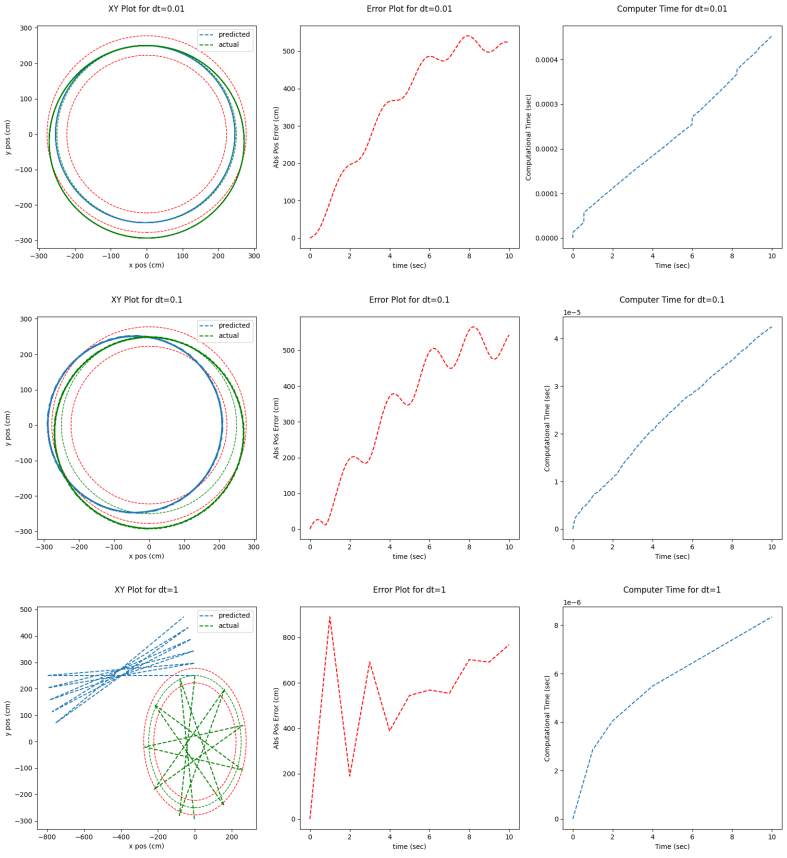


Because a given math command (add, mult, trig) will take a different amount of time based on hardware, calculating the theoretical number of "flops" is completely arbitrary and borderline useless. Therefore, we simply time how long it takes for the computer to do the discretized equations of motion and graph the accumulating time cost.

(d) (10 points) **Graduate Question** Compute part c again where road frictions are much lower (due to rain or ice). Slip causes your tires to respond very differently. Assume that your theta is $\theta_{actual} = \theta(1 - 0.08)$ and velocity is $v_{actual} = v(1 - 0.04)$.

The link to the responsible code, as well as all figures are below:
Link: `https://github.com/ColtonChill/cs6510-midterm/blob/dev/problem01/part_c.py`

Here, we can see a dramatic difference in the actual verses expected position, where actual position grows more and more out of sync with the expected position.



Again, because a computation time in "flops" is arbitrary, we simply time the execution of the motion calculations.

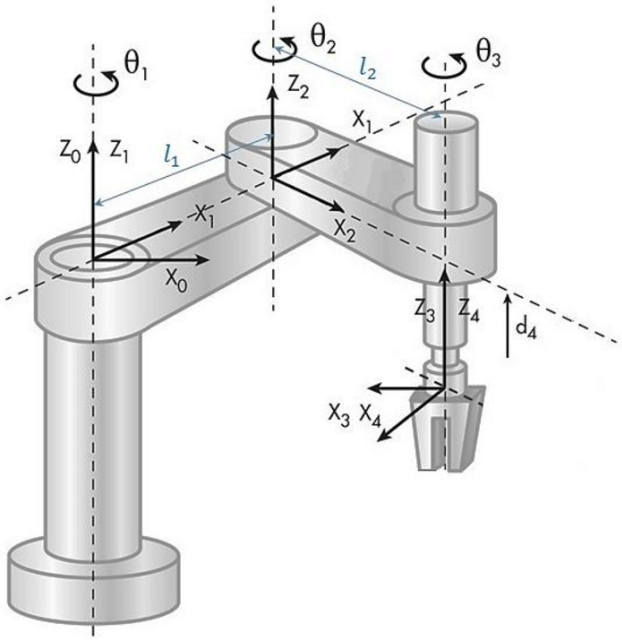


Figure 1: SCARA type manipulator.

2. (20 points) Inverse Kinematics, numerical approaches

   The following SCARA-type manipulator has physical characteristics as follows: $l_1 = 60cm$, $l_2 = 40cm$, and an arbitrary height (ie: assume that $(0, 0)$ is at the first motor).

   (a) (5 points) What is the workspace volume for this robot?. (Draw a picture of it as well)

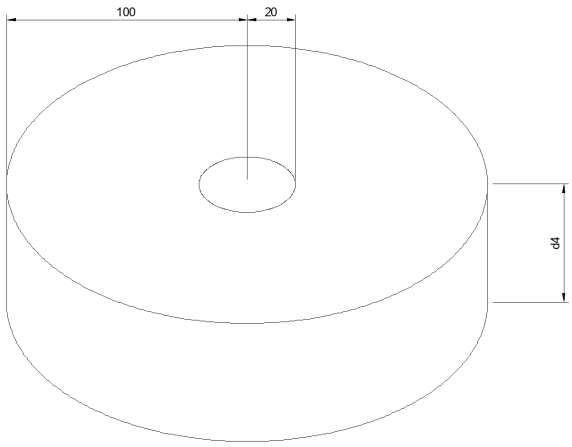


Figure 2: SCARA Volume (cm).

The workspace volume of the robot is: $d_4\pi(100 - 20)^2 = 20,106.19cm^2 d_4$. This remains relative as d_4 is not given.

   (b) (5 points) Write the DH parameters

$joint$	θ	d	a	α
1	θ_1	0	$60cm$	0
2	θ_2	0	$40cm$	0
3	θ_3	d_4	0	0

(c) (10 points) Compute the forward kinematics to find the position of the end effector if $\theta_1 = 30deg$, $\theta_2 = 45deg$, $\theta_3 = 90deg$, and $d = 14cm$

`https://github.com/ColtonChill/cs6510-midterm/tree/master/problem02`

$$A = \begin{bmatrix} c_{30^\circ} & -s_{30^\circ}c_0 & s_{30^\circ}s_0 & 60c_{30^\circ} \\ s_{30^\circ} & c_{30^\circ}c_0 & -c_{30^\circ}s_0 & 60s_{30^\circ} \\ 0 & s_0 & c_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} c_{45^\circ} & -s_{45^\circ}c_{0^\circ} & s_{45^\circ}s_{0^\circ} & 40c_{45^\circ} \\ s_{45^\circ} & c_{45^\circ}c_{0^\circ} & -c_{45^\circ}s_{0^\circ} & 40s_{45^\circ} \\ 0 & s_{0^\circ} & c_{0^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} c_{90^\circ} & -s_{90^\circ}c_{0^\circ} & s_{90^\circ}s_{0^\circ} & 0c_{90^\circ} \\ s_{90^\circ} & c_{90^\circ}c_{0^\circ} & -c_{90^\circ}s_{0^\circ} & 0s_{90^\circ} \\ 0 & s_{0^\circ} & c_{0^\circ} & 14 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ABC = \begin{bmatrix} -0.97 & -0.26 & 0 & 62.31 \\ 0.26 & 0.97 & 0 & 68.64 \\ 0 & 0 & 1 & 14 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$position(x, y, z) = (62.31, 68.64, 14)$$

3. (30 points) Inverse Kinematics with numerical approaches (Use the manipulator on the following page). Note that d_6 refers to the length of the arm on the top and d_1 is the length of the system from the base to the first motor.)

   (a) (10 points) Compute the joint angles and extension distances that will get this robot to pick up an object at x=1.2, y=0.8, z=0.5 if the robot started at if the robot started with the end effector at $\theta_1 = -90deg$, $d_2 = 0.5m$, $d_3 = 1.0m$, $\theta_4 = -90deg$, $\theta_5 = 90deg$, $\theta_6 = 40deg$, $d_6 = 0.2m$.

   For this problem we used a simulated annealing algorithm. The model uses an array of length 500. Each position in the array represents a single move for a single actuator within the system. The array starts out with all movements being random, and the system fills 30 random positions with new random moves each cycle. Telescoping connections (d_2 and d_3) are limited between $-1cm$ and $+1cm$, while rotating joints can be assigned $-1°$, $0°$, or $+1°$. Our system usually gets within $5cm$ of the solution. An example of one particular run is given in the following shell output. Note that it shows final change in angle, not the final angle of that joint. So the final angle of t_1 was actually $-90° + 33° = -57°$. The cumulative motion is a summation of all the motion of the joint before settling. For this part of the problem, the penalty is only the end distance from the target times 10.

```
> python3 a.py
Start Position: (1.06, 0.00, 0.70)
Start Distance: 83.70cm
End Position: (1.200, 0.799, 0.498)
End Distance: 2.51cm
T1(Cumulative/Final): (67°/33°)
T4(Cumulative/Final): (63°/-3°)
T5(Cumulative/Final): (41°/1°)
T6(Cumulative/Final): (42°/-2°)
D2(Cumulative/Final): 46.93cm / -20.20cm
D3(Cumulative/Final): 50.29cm / 38.54cm
```

   (b) (10 points) What would be the joint angles and extension distances to get to the same goal coordinates if the robot started with the end effector at $\theta_1 = -90deg$, $d_2 = 0.5m$, $d_3 = 1.0m$, $\theta_4 = -90deg$, $\theta_5 = 90deg$, $\theta_6 = 40deg$, $d_6 = 0.2m$ and we wished to minimize the total distance traveled for each actuating part?

   For this part of the problem we used the same model. In addition to the penalty from part (a), we added the cumulative motion of all actuators. The resulting shell output is give.

```
> python3 a.py
Start Position: (1.06, 0.00, 0.70)
Start Distance: 83.70cm
End Position: (1.199, 0.802, 0.506)
End Distance: 0.68cm
T1(Cumulative/Final): 51° / 31°
T4(Cumulative/Final): 39° / -13°
T5(Cumulative/Final): 31° / 1°
T6(Cumulative/Final): 38° / -4°
D2(Cumulative/Final): 41.68cm / -18.85cm
D3(Cumulative/Final): 64.53cm / 38.39cm
```

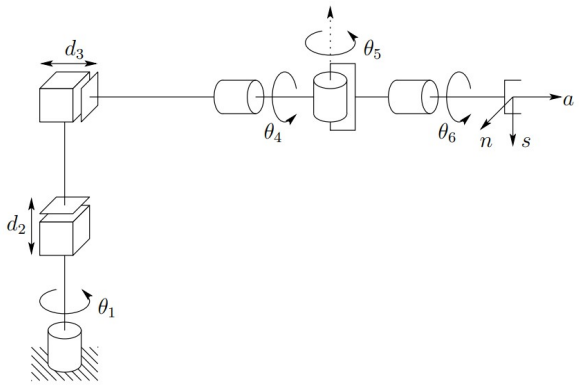
   We can see that as a result of the additional penalty our cumulative changes are diminishing. That being said our result is even better than the results from part (a). The best the system could possibly do while set up in this way, would be for all of the cumulative values to converge with their respective final values.

(c) (10 points) **Graduate Question:** Using the same system, start configuration, and goal coordinates, compute the energy-efficient joint angles and extension distances. For your energy model, assume that actuating θ_1 costs 3x the energy of $\theta_{4,5,6}$, and $d_{2,3}$ costs 2x the energy.

For this part of the problem we used the same model as before. This time, however, we multiplied the appropriate weights to the prescribed cumulative values before adding them to the penalty. The resulting output is given.

```
> python3 a.py
Start Position: (1.06, 0.00, 0.70)
Start Distance: 83.70cm
End Position: (1.188, 0.793, 0.498)
End Distance: 1.40cm
T1(Cumulative/Final): 31° / 29°
T4(Cumulative/Final): 43° / -23°
T5(Cumulative/Final): 44° / -2°
T6(Cumulative/Final): 56° / 18°
D2(Cumulative/Final): 37.49cm / -18.59cm
D3(Cumulative/Final): 55.35cm / 36.68cm
```

As expected, the cumulative values of t_1, d_2, and d_3 are reduced. The other joints are adjusted accordingly to reach the target.



$$
\begin{aligned}
T_6^0 &= \begin{bmatrix} c_1 & 0 & -s_1 & -s_1 d_1 \\ s_1 & 0 & c_1 & c_1 d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 & c_4 s_5 d_6 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 & s_4 s_5 d_6 \\ -s_5 c_6 & s_5 c_6 & c_5 & c_5 d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.34}
$$

$$
\begin{aligned}
r_{11} &= c_1 c_4 c_5 c_6 - c_1 s_4 s_6 + s_1 s_5 c_6 \\
r_{21} &= s_1 c_4 c_5 c_6 - s_1 s_4 s_6 - c_1 s_5 c_6 \\
r_{31} &= -s_4 c_5 c_6 - c_4 s_6 \\
r_{12} &= -c_1 c_4 c_5 s_6 - c_1 s_4 c_6 - s_1 s_5 c_6 \\
r_{22} &= -s_1 c_4 c_5 s_6 - s_1 s_4 s_6 + c_1 s_5 c_6 \\
r_{32} &= s_4 c_5 c_6 - c_4 c_6 \\
r_{13} &= c_1 c_4 s_5 - s_1 c_5 \\
r_{23} &= s_1 c_4 s_5 + c_1 c_5 \\
r_{33} &= -s_4 s_5 \\
d_x &= c_1 c_4 s_5 d_6 - s_1 c_5 d_6 - s_1 d_3 \\
d_y &= s_1 c_4 s_5 d_6 + c_1 c_5 d_6 + c_1 d_3 \\
d_z &= -s_4 s_5 d_6 + d_1 + d_2.
\end{aligned}
$$

Figure 3: Cylindrical/Prismatic manipulator. Use for problem 6.

4. (20 points) Balancing a Pole

   A cart and pole suddenly appears before you and you feel compelled to answer burning questions you have always held about these systems.

   (a) (5 points) Describe the equations of motion that governs this system

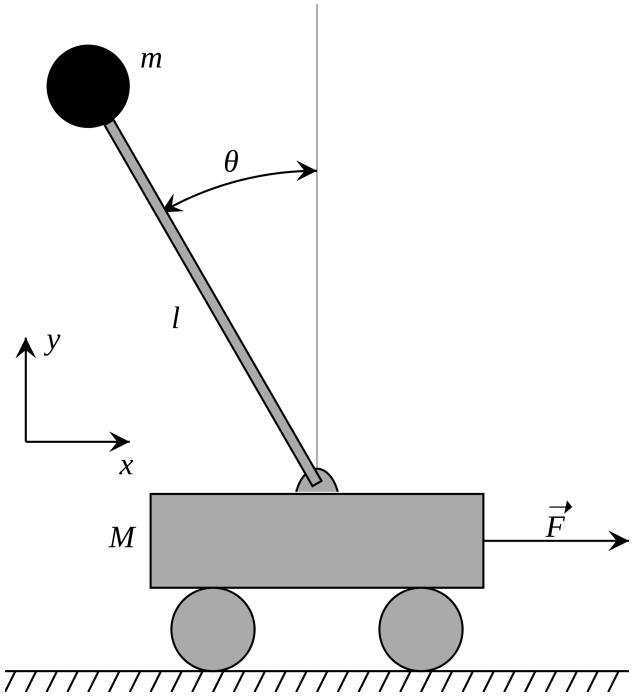


Figure 4: Assume that the mass $m = 0.2kg$, $l = 1.0$, $M = 4.0kg$

The below equations of motion for this problem come courtesy Razvan V. Florian's paper, "*Correct equations for the dynamics of the cart-pole system*".

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

$$\ddot{x} = \frac{F + ml(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta)}{M+m}$$

The other variables $(x, \theta, \dot{x}, \dot{\theta})$ are all input variables for the simulation and don't have simple equations they can be represented with.

(b) (10 points) Generate code for a controller that would be able to keep the pole balanced in the air.

The below link is to a Neural Net Q-learning Agent we implement to solve the car-pole problem.
Link: `https://github.com/ColtonChill/cs6510-midterm/tree/master/problem04`

(c) (5 points) What is the maximum angle that my pole can fall to before it cannot recover if max Force $F = 6N$

For this problem, we need to assume some initial parameters to constrain the infinite number of possible solutions. Here we are assuming, for simplicity, $\dot{x} = 0$ and $\dot{\theta} = 0$, where x is irrelevant and

$\ddot{x}$ is a constant scalar base of the constant $F = 6N$.

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

$$0 = \frac{9.8\sin\theta + \cos\theta\left(\frac{-6}{4.2}\right)}{\frac{4}{3} - \frac{0.2\cos^2\theta}{4.2}}$$

$$0 = 9.8\sin\theta - \frac{6}{4.2}\cos\theta$$

$$\frac{6}{4.2} = 9.8\tan\theta$$

$$\boxed{\theta = 0.145\ rad = 8.294^\circ}$$

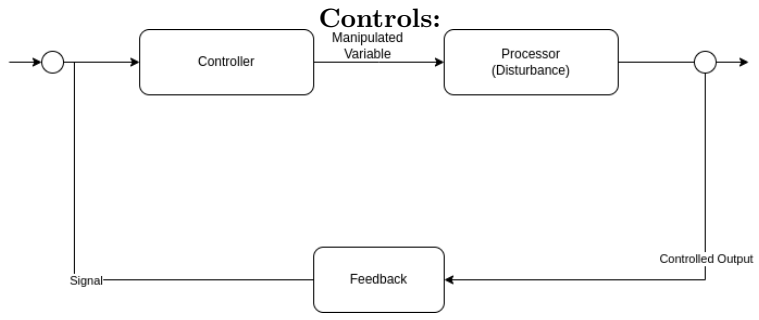
This matches up with experimental results in the simulation which find the farthest possible theta the pole can fall to be in between $\theta = 8.2^\circ \rightarrow 8.3^\circ$

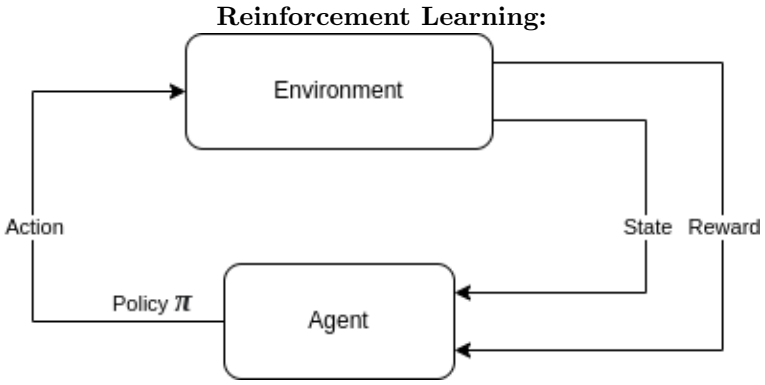
5. (30 points) Control and Reinforcement Learning

   (a) (5 points) Explain three merits and three demerits of using reinforcement learning for mechatronic systems.

- Merit: Reinforcement Learning works really well when we want the problem to be explored or to find many different options/routes to solve the problem.
- Merit: It works well even without having a specific target to hit since it will act in a way to maximize its reward.
- Merit: Errors that occur during training can be overcome and unlearned.
- Demerit: Reinforcement learning takes a lot of computation and time.
- Demerit: Reinforcement only really shines when a sequence of actions can lead to a given outcome. If the problem lacks that distinct sequence, reinforcement learning becomes a lot more difficult
- Demerit: It can be slow to learn given a bad start on its environment. E.g. it can encounter a lot of local minima in its data and decision trees.

   (b) (5 points) Draw a diagram for reinforcement learning and controls and contrast the two





The big difference between Controls and Reinforcement Learning is the lack of input and output on the side of Reinforcement Learning. The Reinforcement Learning agent doesn't start with any input, as it starts from ground zero, knowing nothing about it's environment. It also simply keeps going and improving, not waiting until it reaches some predetermined output (unless specifically programmed by the user).

Another key difference is the role the environment plays in the training of the agent. The environment allows the agent to make various decisions and act and learn different policies given the state; whereas

controls can only react against the most recent feedback. This allows reinforcement learning models to learn and succeed in many various and often complicated systems, such as games. A great example is AlphaGo, which was able to defeat the world's best Go player.

(c) (10 points) Use OpenAI Gym to create a trained reinforcement learning model for the cart and pole problem (CartPole-v1).

`https://www.gymlibrary.ml/environments/classic_control/cart_pole/`

This tutorial will help you get started with using the OpenAI gym python library. `https://blog.paperspace.com/getting-started-with-openai-gym/`

`https://github.com/ColtonChill/cs6510-midterm/tree/master/problem05/c.py`

(d) (10 points) **Graduate Question:** Implement this example of a 2D running robot (cheetah) and adapt the code to penalize hip motor movements faster than $\pi \frac{rad}{s}$. `https://github.com/openai/gym/blob/master/gym/envs/mujoco/half_cheetah.py` A Virtual Machine that was made with VMWare is available on Canvas for you to use for this question if you have trouble installing Mujoco.

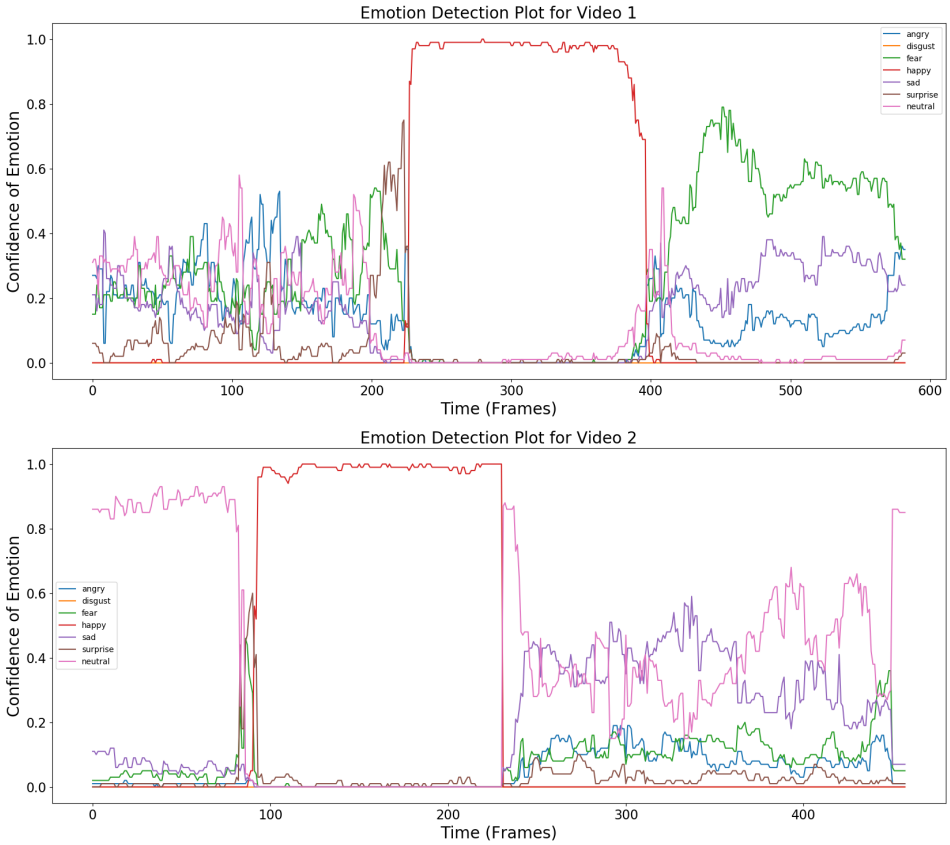
`https://github.com/ColtonChill/cs6510-midterm/tree/master/problem05/d.py`

6. (30 points) Human Emotions

   (a) (10 points) Using the FER library in python, example in the following link: `https://towardsdatascience.com/the-ultimate-guide-to-emotion-recognition-from-facial-expressions-using-python-64e58d4324ff` build a system that classifies human emotions and validate on video 1 and 2 from this repository: `https://github.com/rjrahul24/ai-with-python-series/tree/main/07.%20Emotion%20Recognition%20using%20Live%20Video` Generate plots of predicted emotions over time for both videos.

Below is a link to the classifier used to validate videos 1 & 2.

Link: `https://github.com/ColtonChill/cs6510-midterm/tree/master/problem06`

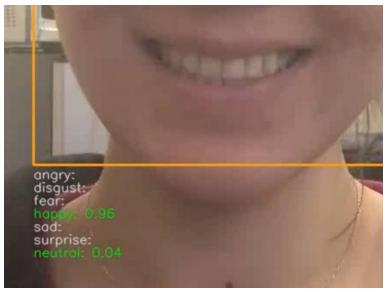


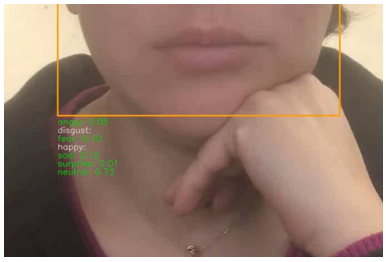
   (b) (5 points) Do the same as the above with a video feed from your webcam. Set your software up to allow video feed or a pre-recorded video. (In essence, make faces at yourself and make sure that your service works). Submit a short faces-recording along with the script that can read live webcam streams.

Below is a link to the scrip used to classified our webcam video feeds.

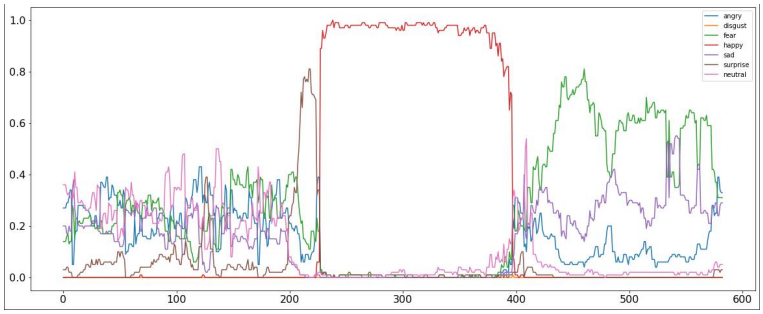
Link: `https://github.com/ColtonChill/cs6510-midterm/blob/master/problem06/b.py`

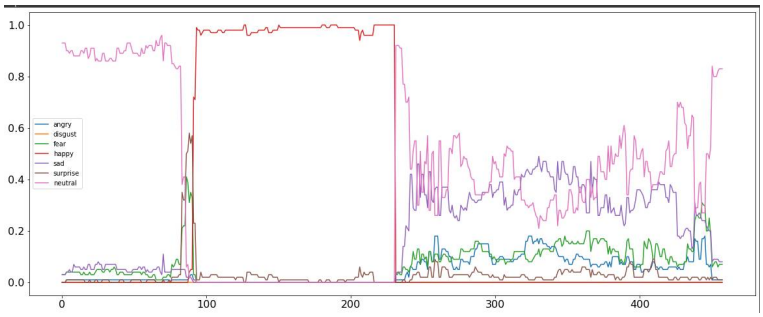
Below are screenshots of the emotional classifier in action.





Lastly the time series graphs of the classifier's evaluation confidence.





	Human Emotions	Emotion Value from the Video
0	Angry	20.80
1	Disgust	0.00
2	Fear	32.58
3	Happy	137.37
4	Sad	76.43
5	Surprise	13.04
6	Neutral	177.10

(c) (5 points) What are the logical applications of this tool for an autonomous robot? What are the

ethical and legal consequences of fielding a system that makes decisions based on this tool?

Provision of personalised services

- analyze emotions to display personalised messages in smart environments
- provide personalised recommendations e.g. on music selection or cultural material
- analyze facial expressions to predict individual reaction to movies

Customer behaviour analysis and advertising

- analyze customers' emotions while shopping focused on either goods or their arrangement within the shop
- advertising signage at a railway station using a system of recognition and facial tracking for marketing purposes

Healthcare

- detect autism or neurodegenerative diseases
- predict psychotic disorders or depression to identify users in need of assistance
- suicide prevention
- detect depression in elderly people

Employment

- help decision-making of recruiters
- identify uninterested candidates in a job interview
- monitor moods and attention of employees

Education

- monitor students' attention
- detect emotional reaction of users to an educative program and adapt the learning path
- design effective tutoring system

Public safety

- lie detectors and smart border control
- predictive screening of public spaces to identify emotions triggering potential terrorism threat.
- analyzing footage from crime scenes to indicate potential motives in a crime

Crime detection

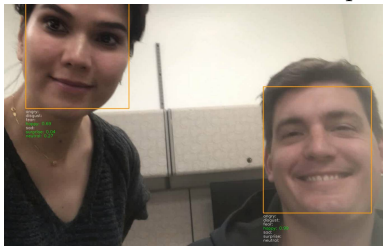
- detect and reduce fraudulent insurance claims
- deploy fraud prevention strategies
- spot shoplifters

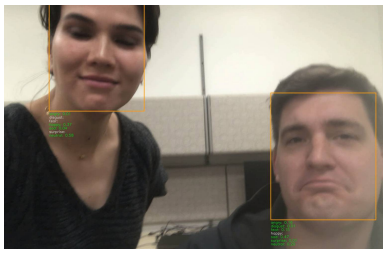
Other

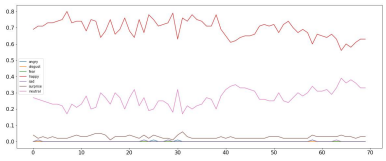
- driver fatigue detection
- detection of political attitudes

(d) (10 points) **Graduate Question:** Set up a service that allows 2 or more faces to be processed at once.

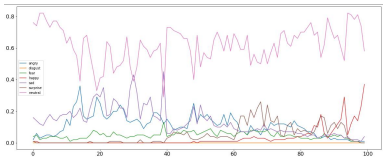
The code linked to in part (b) is also capable of multi-face/emotion detection. See the 'REDME.md' file on GitHub for links to videos of these processes running in real time.







	Human Emotions	Emotion Value from the Video
0	Angry	0.05
1	Disgust	0.00
2	Fear	0.02
3	Happy	48.58
4	Sad	0.05
5	Surprise	1.93
6	Neutral	18.86



	Human Emotions	Emotion Value from the Video
0	Angry	11.35
1	Disgust	0.00
2	Fear	4.76
3	Happy	3.11
4	Sad	13.44
5	Surprise	5.08
6	Neutral	62.15

7. (30 points) Motion Planning

   (a) (20 points) Implement an A* based planner and compare it's results with the Djikstra, A-Star, RRT-Star, Bi-Directional A-Star, and the Breadth First Search Planners from this github. `https://github.com/AtsushiSakai/PythonRobotics/tree/master/PathPlanning`. Create a table comparing the average cost of the path found over 10 iterations along with the time to convergence.

Planner	Path length	Calc. time (ms)
My A-Star	95.0	770.2510000000001
A-Star	100.0	114.8868
BFS	100.0	51.2005
Bidirectional A-Star	102.0	119.87179999999998
Djikstra's	100.0	50.558499999999995
RRTStar	167.8	383.40209999999996

After you have implemented your planner and compared it answer the below questions.

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

  The lowest cost for a path was (surprisingly) found by my own A-Star Planner, followed by a 3-way tie between the other A-Star, Breadth First Search, and Djikstra's planners.

- Which one found a path the fastest on average?

  The fastest algorithm was Djikstra's, being just faster than Breadth First Search. I suppose that this is because no matter the algorithm, nearly all of the spaces in the maze were traversed anyway, making A-star a less appealing option, as it has some overhead with calculating the heuristic and picking a "best node". It also is not quite as exhaustive as BFS, so I assume that's why it just edged it out.

- 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?

  The planner that I designed based on the information in the provided article was considerably slower than any of the provided algorithms. I conjecture that a big reason for why this is, is that the other planners use a dictionary to store their closed nodes, and finding an element in a dictionary is a lot faster than finding an element in a list, like what I did, and having that difference in computing time 1000 times will certainly lead to a considerable amount of overhead.

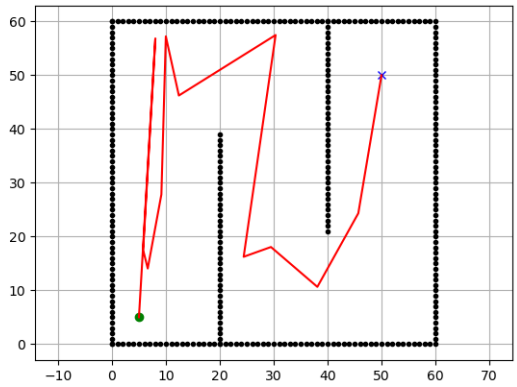
- Which planner appears to be the best overall? Which planner would you use for a robot in a complex environment?

  Right now, it definitely looks like Djikstra's algorithm is the best overall, but not by a huge margin. BFS also performed quite well in the maze, but that would definitely not work well in a complex

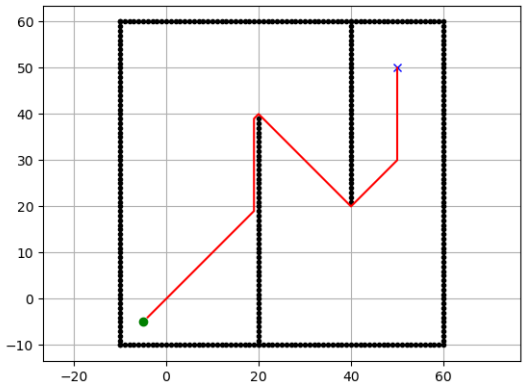
environment, since it's really exhaustive and tracks every node given. I think in a complex environment, A-Star would likely perform the best, since it tends to ignore unnecessary nodes. It picks the "best" one each iteration, which is something Djikstra's doesn't do.

(b) (10 points) **Graduate Question:** Implement a skid-steer (4 wheel) vehicle model for a robot 33 inches wide, by 40 inches long for the Hybrid A* algorithm. Change the scenario so both your A* and Hybrid A* have the same obstacle field. Compare and contrast the final trajectories, graph these. Write what the algorithmic differences are between the two planners and how the kinematics are used to constrain the Hybrid A*.

Planner	Path Length	Time
Hybrid	20.6	652.8015
Normal	29.9	439.661



Here is a result of the Hybrid Sampling-Based A-Star algorithm:



Here is a result of the regular Sampling-Based A-Star algorithm:

The similarities between the A* and Hybrid A* algorithms are pretty small. The main similarity is that both algorithms pick their next move based on a heuristic calculation, which is the estimated distance from any node to the goal node. The Hybrid A* algorithm calculates this based on continuous (not grid-based) space.

The A* algorithm changes quite a bit when it is turned into a sampling-based pathfinding algorithm. Random sampling will likely not ever yield the very shortest possible path, as we can see in the difference between graphs. The variance between run-times and computational complexity also vary considerably between different runs. However, sampling-based algorithms are pretty easy on computing power. A lot of the time overhead that came from the hybrid algorithm comes from sifting through random samples and generating new ones until a valid sample can be used.

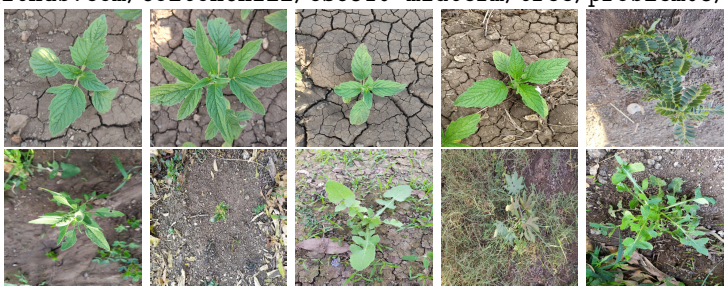
The kinematics of the skid-steer model are mostly used to determine collisions between the vehicle and the obstacle space (or any space that isn't traversable). This proved to be a somewhat difficult geometric challenge. Skid-steer type robots change their angle as they travel and collisions needed to be calculated not only when the vehicle's axes were parallel to the space's, but also when the vehicle was at an angle relative to the space. The vehicle model was also used to generate paths between the nodes/samples.

8. (30 points) Object Detection (Please put the classified images as part of your submission)

   (a) (5 points) Classify the first 10 pictures in `https://github.com/ravirajsinh45/Crop_and_weed_detection` using any image classification algorithm.

Below is a link to our implementation of a generic CNN classifier to sort images of crops vs. weeds.

Link: `https://github.com/ColtonChill/cs6510-midterm/tree/problem08/problem08/`

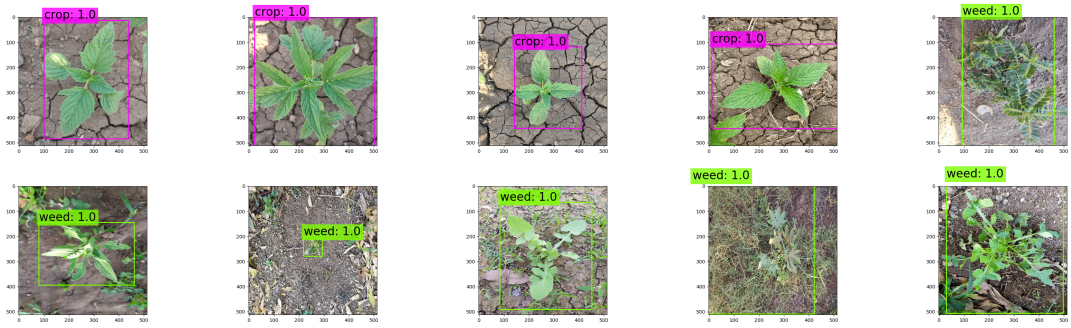


These 10 images can be sorted by following the above link and running the "part_a_demo.py" code. The training takes approximately 20 epochs to consistently get 100% accuracy.

   (b) (5 points) Implement Yolo and do the same, what are the differences.

Below is a link to our implementation of a YOLO classifier and it's outputs on the same 10 images.

Link: `https://github.com/ColtonChill/cs6510-midterm/blob/dev/problem08/part_b_demo.py`



As one can see, the YOLO classifier draws a bounding box around the object that it detects rather then simply picking up on any patterns found in the image (such a background). This results in YOLO having a higher successes rate and better general object detection.

   (c) (10 points) Using transfer learning, pick an image classification algorithm and retrain it to learn to detect a new object of interest.

The below link is our implementation of Alex-net which we have retrained to detect a new object. Alexnet is a CNN which is pretrained to on Imagenet `www.image-net.org/`, a massive database of 1,000 different classes of object, ranging from banjos to moray eels. The categories of "weed" and "crop" are not recognized classes in imagenet. For simplicity's sake, we have chosen to retrain Alexnet on the same above dataset of crops vs. weeds. Below is the link to this implementation:

Link: `https://github.com/ColtonChill/cs6510-midterm/tree/problem08/problem08`

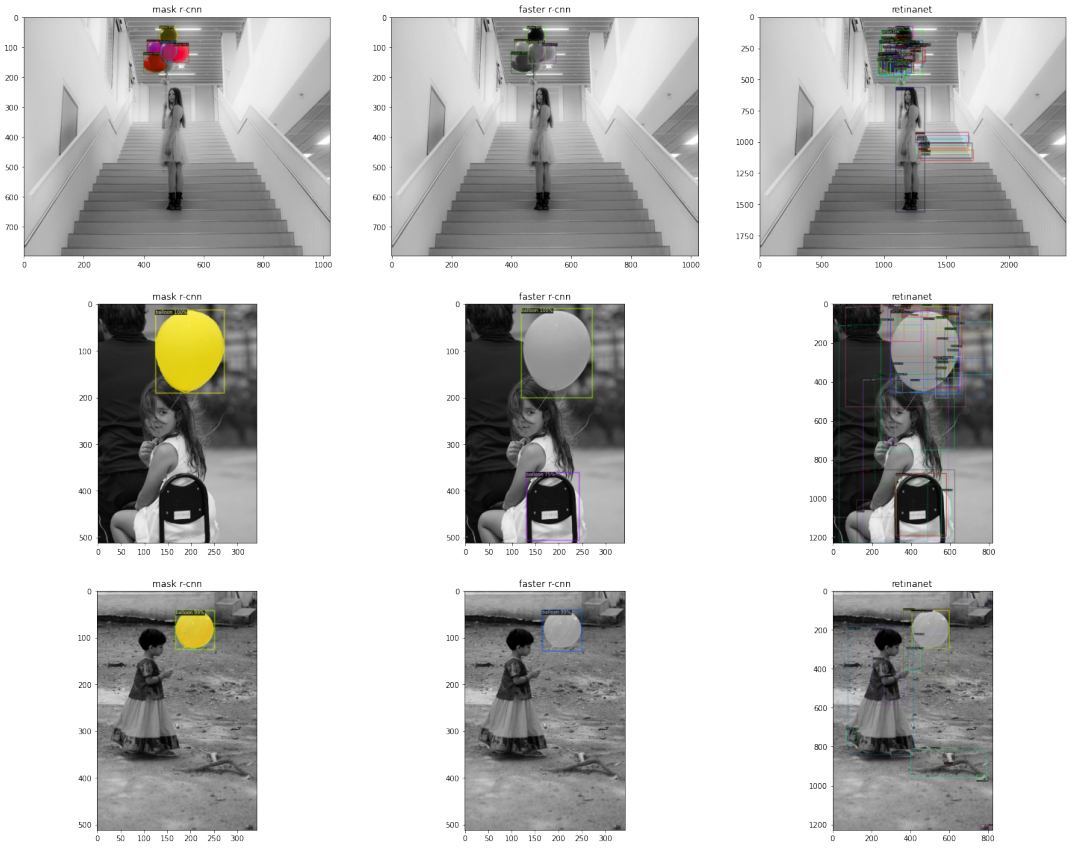
By running the "part_c" code snippets to train and demo both a pretrained and untrained Alexnet, the user will be able to witness the reduced number of epochs necessary to fine-tune Alexnet for the new dataset. We find the model training to reduce from 10 epochs to 2 with learning transfer.

   (d) (10 points) **Graduate Question:** Implement Detectron (`https://github.com/facebookresearch/`

`Detectron`) and test the image segmentation of the objects you are interested in. Give metrics on relative segmentation/classification quality comparing: Mask R-CNN, faster R-CNN, and RetinaNet.

In order to implement and test Detectron2, a GPU is needed. Therefore, we had to implement this solution on google colab. The link to said notebook and new object dataset can be found below.
Link: `https://drive.google.com/drive/folders/1y1VXsz2n4byxgcOcAXxnSQOV1X1rgsS9?usp=sharing`

For the new object class to train the R-CNNs on, we choose a extensive party balloon dataset in YOLO format. We retrained 3 stock models (Mask, Faster, and FetinaNet R-CNN) to detect the balloon object. Below is a visualization of the outputs of each of these models on 3 random images from the dataset.



Below are the testing metrics of each of these models over the entire testing set.

Metric	Mask R-CNN	faster R-CNN	RetinaNet
AP	72.353	66.872	74.573
AP50	83.93	83.953	87.853
AP75	80.06	83.953	83.185
APs	0.0	0.0	30.297
APm	53.85	53.972	60.919
APl	89.159	81.116	88.706

As can be seen above, the mask R-CNN has overall slightly better performance then the faster R-CNN. Though the training time for the faster R-CNN is considerably better. The RetinaNet continues to output the vestigial classes it was previously trained on resulting in a cluttered visualization. However, it is still able to isolate the balloon to a slightly better degree then each of the other nets.

9. (20 points) Ethics of Robotics - Open Ended Questions
   (a) (5 points) Many people (particularly those in the robotics industry) believe that robotics is purely within the purview of technical development and should not have any ethical considerations. What do you feel can be a merit or demerit to this way of thinking?

No, unless we consider a robot to be a highly developed washing machine. The answer would be less obvious if robots had the capacity for emotion or feeling. Marvin Minsky, one of the pioneers of AI, held that there is no fundamental distinction between humans and machines and that the development of universal AI is impossible without the inclusion of self-aware emotions in robots. In the discussion around robot rights, it has been suggested that while giving robots the freedom to exist and carry out their tasks, this freedom should be contingent on their need to serve humans. There is a great deal of debate surrounding this topic.

   (b) (5 points) Isaac Asimov listed 3 laws of robotics, comment on the algorithmic complexity of implementing these into working intelligence. Define a scenario and write Psuedo code to implement these rules.

In my opinion there is no robot or, in the loose sense of the term, automatic machine that exists to which any of them apply. Because computers and the machines under their control carry out instructions exactly as planned, if they were told to injure humans or come into contact with them while doing so, they will do it out of necessity. On the surface, these concepts sound reasonable. However, they are based on concepts developed by humans. (Just an interesting point : Chris Stokes, a philosopher at Wuhan University in China, says the First Law fails due to language ambiguity and complex ethical dilemmas that cannot be resolved with a simple yes or no. Because it is immoral to have a law that forces sentient beings to continue living as slaves, the Second Law is invalidated. The Third Law is ineffective because it creates a persistent social stratification and this legal framework contains a tremendous amount of possibilities for exploitation.) We consider that we have an environment in which there is either a human or a danger, or both or neither. And we have a robot that wants to execute a series of instructions according to the programming(fire, stop, retreat or advance...). It is worth mentioning that in this scenario we have a hierarchical view here because we want our robot to act according to three laws of Isaac Asimov. And of course, we will find out later that the full implementation of these laws is not possible in some cases and they conflict with each other. (Regardless of details, robot's motors allow for movement with the help of its camera and object-detection model, it can detect its surroundings and etc).

Psuedo code to implement these rules:

if human visible = yes and danger visible = yes and command = none Then

action = does not fire first law

if human visible = no and danger visible = " and command = any Then

action = obeys command second law

if human visible = no and danger visible = yes and command = none Then

action = shoot third law

if human visible = yes and danger visible = yes and command = shoot Then

action = does not fire first law > second law

if human visible = yes and danger visible = yes and command = none Then

action = does not fire first law > third law

if human visible = ” and danger visible = yes and command = retreat Then

action = retreat second law > third law

(c) (5 points) In the event of an autonomous system causing harm or damages, who is responsible? Read and comment on the following two documents: `https://www.callahan-law.com/articles-and-expert-advice/when-an-autonomous-vehicle-hits-a-pedestrian-who-is-responsible/` and `https://en.wikipedia.org/wiki/Self-driving_car_liability`

It is not always easy to identify who or what is at fault in an accident involving an autonomous vehicle. Regular cars don't employ the wide range of software and sensors that autonomous vehicles utilize. I believe that the makers of sensor systems and software developers should be held liable in the event that something goes wrong. The difficulty in figuring out how the various algorithms can detect every conceivable situation on the road has considerably slowed the development of self-driving automobiles. For instance, if a child's ball rolls out onto the road, the autonomous car needs rely on complex data and algorithms to "see" the object and swerve. A driver who is human will make decisions and make them quickly. A split-second decision to hit the ball instead of swerving into oncoming traffic would have a different result in a self-driving automobile.

(d) (5 points) What laws may be helpful for regulating or controlling autonomous systems? What drawbacks will this potentially have?

Regulating something you don't completely comprehend is challenging. Since autonomous technologies are still in their infancy it is difficult, if not impossible, to foresee scientific breakthroughs in this area in the near future. Any new rules and legislation must be sufficiently broad to avoid being quickly out of date while also not being so ambiguous as to offer no useful guidance. Due to the complexity of this task neither global international treaty on autonomy nor any domestic legislative actions governing autonomous technologies will be established in the near future. Instead, discrete laws governing comparatively narrow facets of autonomous technologies may be what we should expect.