EE3EY4: Electrical Systems Integration Project
Lab 8: Driver-Assist Collision Avoidance & Autonomous Driving Using Wall-Following

Instructor: Dr. Sirouspour

Aithihya Kompella - kompella - 400310794
Benji Richler - richlerb - 400296988
Shathurshika Chandrakumar - chands39 - 400315379
W. Tisuka Perera - L03 - pererw2 - 400318373

Date of Submission: April 5th, 2023

Part 1: Driver-Assist Collision Avoidance Algorithm

1. **Task: Explain why this might be a reasonable assumption based on the configuration of the sensors on MacAEV.**
   This may be a reasonable assumption based on the configuration of the LiDAR, IMU and camera because the laser frame can now be ignored. The LiDAR's orientation with regards to the lase_link frame allows this to be possible. In addition, the IMU and camera can both accurately estimate the orientation and motion of the MacAEV, which means that the laser frame is no longer required. The relative orientation between the laser and the base_link can also be estimated by the fusion algorithm that is present in the code. Therefore, the laser frame is not required.

2. **Task: With this assumption, the angle α_i can be easily computed from the angle information in scan data. Explain how this is done.**
   The LiDAR scan data function does this. The LiDAR is aligned with the base_link frame, and the range and angle are given with regards to the corresponding LiDAR frame, which means it is possible to use angle information to calculate the angle α_i. This can be done by adding the angle of the laser beam with respect to the LiDAR frame and the yaw angle of the base_link frame.

3. **Task: Derive the above expression for δ_o^{k+1}. Also, explain the role of the design parameter $0 \leq \eta_\delta \leq 1$.**
   Repelling Force Steering Angle Contribution Derivation:

$$\eta_\delta(\text{joystick acceleration}) + (1-\eta_\delta)(\text{repulsive force}) = \text{total steering acceleration}$$

$$\eta_\delta \frac{V_s^2}{L}\tan\delta_{Joy}^{k+1} + (1-\eta_\delta) f_{oy} = \frac{V_s^2}{L}\tan(\delta_{Joy}^{k+1} + \delta_o^{k+1})$$

$$\eta_\delta \tan\delta_{Joy}^{k+1} + (1-\eta_\delta)\frac{L}{V_s^2} f_{oy} = \tan(\delta_{Joy}^{k+1} + \delta_o^{k+1})$$

$$u_o = \tan(\delta_{Joy}^{k+1} + \delta_o^{k+1}) \qquad \tan^{-1} u_o - \delta_{Joy}^{k+1} = \delta_o^{k+1}$$

$$\delta_o^{k+1} = \tan^{-1} u_o - \tan^{-1}(\tan\delta_{Joy}^{k+1}) = \tan^{-1}\left(\frac{u_o - \tan\delta_{Joy}^{k+1}}{1 + u_o \tan\delta_{Joy}^{k+1}}\right)$$

   The main role of the y-acceleration formula is to transform the repulsive force f_{oy} into the steering angle contribution, which can be performed by the vehicle itself. η_δ is the weighting factor that determines the repulsive force strength with regarding the the input from the joystick.
   small η_δ = larger f_{oy} steering contribution, large η_δ = smaller f_{oy} steering contribution

4. **Task: Locate the following function in the code:**

```
def preprocess_lidar(self, ranges):

        data=[]
        j=0
        for i in range(self.ls_len_mod):
            if ranges[self.ls_str+i]<=self.max_lidar_range:
               data.append([ranges[self.ls_str+i],i*self.ls_ang_inc-
self.angle_cen])
               j=j+1
        self.ls_len_mod2=j
        return np.array(data)
```

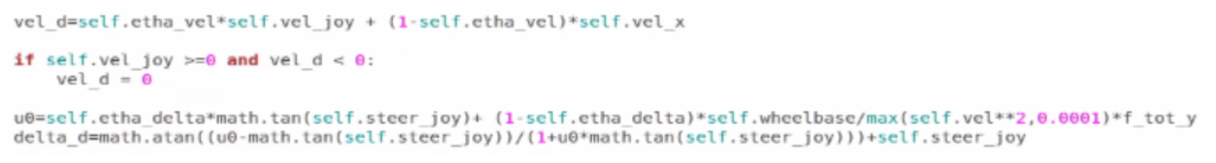
   Explain what this function does.

This function has two parameters: the self-object and “ranges”. Then, an empty array is created called “data”. The variable j is introduced as a counter, followed by a for loop with the variable i, which iterates to the value of “self.ls_len_mod”.

This is an integer that represents the number of scan beams in front. The index of ranges at “self.ls_str + i” is compared to “self.max_lidar_range”. This line is checking if the range of values at the start index at 180 degrees plus the value of the variable i is less than or equal to the maximum range of the lidar. If this is true, then an array containing the (value of ranges at the index i multiplied by the beam angle increment) - (the front center beam angle) is appended to the counter variable j, which increments by 1. The number of laser returns that are inside the maximum valid range is then set to j. Then, the array “data” is returned.

Therefore, what the function does is count how many LiDAR beams are not travelling the maximum range because this indicates that there is an object in the way of the beam that prevents it from travelling. The difference between the angle of the front mean (90) and the angle of the beam that is less than the maximum range is appended to a data array and returned, along with the distance at the degree that the beam is at.

5. **Task: Finally, find and complete the following lines of code in the function “lidar_callback”. Explain the purpose of this part of the code:**



```
vel_d=self.etha_vel*self.vel_joy + (1-self.etha_vel)*self.vel_x

if self.vel_joy >=0 and vel_d < 0:
    vel_d = 0

u0=self.etha_delta*math.tan(self.steer_joy)+ (1-self.etha_delta)*self.wheelbase/max(self.vel**2,0.0001)*f_tot_y
delta_d=math.atan((u0-math.tan(self.steer_joy))/(1+u0*math.tan(self.steer_joy)))+self.steer_joy
```

The first line is based on the formula

$$v_s^{k+1} = \eta_v v_{joy}^{k+1} + (1 - \eta_v) v_{so}^{k+1}$$

which is composed of the weight factor times the joystick input velocity and 1 minus the weight factor times the obstacle-corrected velocity based on the algorithm. When η_v is 1, the vehicle’s velocity will be fully dependent on the input velocity and have no obstacle avoidance assistance. When η_v is 0, the opposite is true; the vehicle’s velocity will be purely based on the obstacle avoidance assistance algorithm and the user’s joysticks input will have no effect on the vehicle’s movement.

The u0 line is based on the formula

$$u_0 = \eta_\delta \tan \delta_{joy}^{k+1} + (1 - \eta_\delta) \frac{l}{(\max(v_s^2, \epsilon))} f_{oy}$$

which represents the corrective sterling angle in which ε = 1*10^-4, a very small number. The weighting factor η_v works the same way as before, with η_v =1 resulting in pure joystick driving and η_v = 0 resulting in pure obstacle avoidance assistance driving.

6. **Task: Examine the code and relate the remaining parameters to those used in the collision avoidance assistance algorithm described above. You should tune these parameters to achieve a desired response from the vehicle. The values given above should serve as a good starting point for your tuning process.**
Drive the vehicle in the virtual environment with and without collision avoidance assistance and notice any difference in its behavior. Use the simulation to explore the impact of the above parameters on the vehicle response. Include your observations in your report.

During the simulation, it was observed that without collision assistance, the car kept colliding with the walls and limits once it was steered in any direction and approached the walls. When collision assistance was activated, the car stopped moving as it approached the wall after a certain turning angle. In this way the functionality of collision assistance was demonstrated.

Part 2: Self-Driving by Following Walls

7. **Task: Some of the derivations that will follow are intentionally left incomplete. You must complete and include these derivations in your report.**

   D_{lr} 2nd Derivation:

$$\dot{d}_{Lr} = \dot{d}_L - \dot{d}_r$$

$$\ddot{d}_{Lr} = \ddot{d}_L - \ddot{d}_r = -\frac{V_s^2}{L}\cos\alpha_r\tan\delta - \frac{V_s^2}{L}\cos\alpha_r\tan\delta = \boxed{-2\frac{V_s^2}{L}\cos\alpha_r\tan\delta}$$

   Steering Angle Derivation:

$$\ddot{d}_{Lr} + k_d\dot{d}_{Lr} + k_p d_{Lr} = k_p d_{Lr}^{des}$$

$$\ddot{d}_{Lr} + k_d\dot{d}_{Lr} + k_p\tilde{d}_{Lr} = 0$$

$$\ddot{d}_{Lr} = -k_d\dot{d}_{Lr} - k_p\tilde{d}_{Lr} = -k_p\tilde{d}_{Lr} - k_d\dot{d}_{Lr}$$

$$\ddot{d}_{Lr} = -2\frac{V_s^2}{L}\cos\alpha_r\tan\delta$$

$$\tan\delta = -\frac{\ddot{d}_{Lr}L}{2V_s^2\cos\alpha_r}$$

$$\delta = \tan^{-1}\left(-\frac{L}{2V_s^2\cos\alpha_r}\ddot{d}_{Lr}\right) \longrightarrow \boxed{\delta = \tan^{-1}\left(-\frac{L}{2V_s^2\cos\alpha_r}\left(-k_p\tilde{d}_{Lr} - k_d\dot{d}_{Lr}\right)\right)}$$

   Closed Loop Transfer Function Derivation:

$$\ddot{d}(s) + k_d\dot{d}(s) + k_p d(s) = k_p d^{des}(s)$$

$$\text{Laplace transform: } s^2 D(s) + s k_d D(s) + k_p D(s) = k_p D^{des}(s)$$

$$D(s)\left(s^2 + sk_d + k_p\right) = k_p D^{des}(s)$$

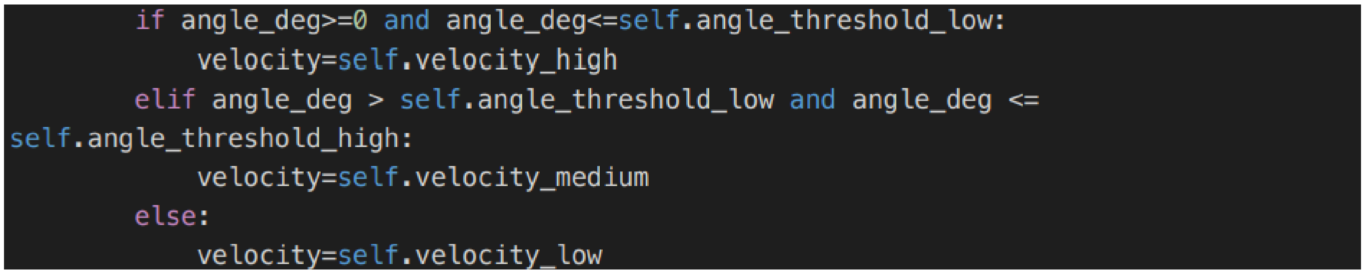
$$\boxed{\frac{D(s)}{D^{des}(s)} = \frac{k_p}{s^2 + sk_d + k_p}}$$

8. **Task: What is the steady-state tracking error in response to a constant desired reference command for the distance to the wall in the closed-loop control system?**

$$\frac{D_{(s)}}{D^{des}_{(s)}} = \frac{\frac{k_p}{s(s+k_d)}}{1+\frac{k_p}{s(s+k_d)}}$$

$$e_{ss} = \left.\frac{1}{1+\frac{k_p}{s(s+k_d)}}\right|_{s=0} = 0$$

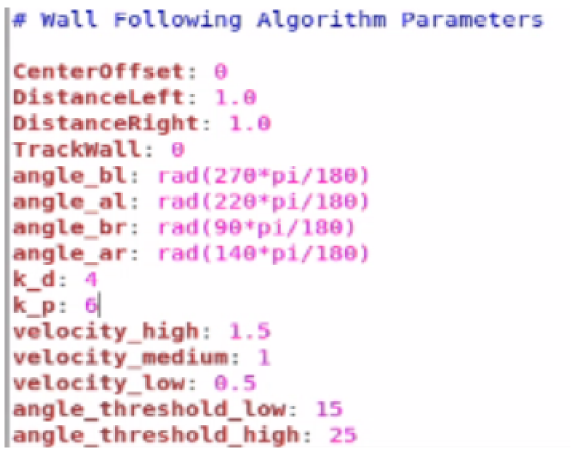
9. **Task: Explain what the following section of the code is trying to accomplish:**



```
        if angle_deg>=0 and angle_deg<=self.angle_threshold_low:
            velocity=self.velocity_high
        elif angle_deg > self.angle_threshold_low and angle_deg <= 
self.angle_threshold_high:
            velocity=self.velocity_medium
        else:
            velocity=self.velocity_low
```

In this section of the code, a velocity control system is being implemented. This is based on the angle of the obstacle that is detected by the LiDAR system. Here, if the variable "angle_deg", which represents angle in degrees, is greater than or equal to zero and less than or equal to the self object's lower angle threshold. If this is the case, then the velocity of the AEV is high. If the variable "angle_deg" is greater than the low angle threshold and less than or equal to the higher angle threshold, then the velocity is medium, and in all other cases the velocity is low. This can be seen when running the AEV, since when it is further from an obstacle, the velocity is high, but as it gets closer to the obstacle its velocity decreases.

10. **Task: The values given above represent a starting point, but you are encouraged to adjust them to tune up of the response of the vehicle. You should explain in your report how you have selected these parameters. Note that you have the option of controlling the vehicle in one of three modes, distance-to-left wall, distance-to-right wall, or an offset from the center position of the walls. You can choose the driving mode by setting the value of the parameter "TrackWall" in the file "params.yaml". Observe the vehicle behavior in the simulation environment in each of these modes and comment on what you see in your response.**


```
# Wall Following Algorithm Parameters

CenterOffset: 0
DistanceLeft: 1.0
DistanceRight: 1.0
TrackWall: 0
angle_bl: rad(270*pi/180)
angle_al: rad(220*pi/180)
angle_br: rad(90*pi/180)
angle_ar: rad(140*pi/180)
k_d: 4
k_p: 6
velocity_high: 1.5
velocity_medium: 1
velocity_low: 0.5
angle_threshold_low: 15
angle_threshold_high: 25
```

The driving mode is set by changing the value of TrackWall to 0, 1 and 2 for the modes distance-to-left wall, distance-to-right wall, and offset from center position respectively. In the distance-to-left wall mode, the vehicle adjusts to maintain a fixed distance, DistanceLeft, from the left wall. Similarly, in the distance-to-right wall mode, the vehicle adjusts its fixed distance to the right wall. In the offset from center position mode, the vehicle adjusts itself to a specified distance from the center, as set by CenterOffset, of the left and right walls detected.

The DistanceLeft and DistanceRight parameters were set to be 1, as this gave an appropriate distance from a single wall that the vehicle will adjust to avoid collisions most effectively. The CenterOffset value was set to 0, so the vehicle is able to center itself exactly in the middle of the left and right walls detected.

The angle_lb/al and angle_br/ar are the angles at which the sensors detect the left and right walls respectively. The velocity_high, velocity_medium and velocity_low parameters were set to 1.5, 1 and 0.5 respectively. These velocities allowed the vehicle to travel in a straight corridor at a reasonable speed, while also being able to navigate sharp turns slow enough that the algorithm is able to avoid collisions. The angle threshold low and high were set to 15 and 25, as these thresholds allowed the vehicle to turn effectively to limit collisions.

11. **Task: Comment on the trade-offs in using each of the three control modes to self-drive the vehicle in this environment. Where do you see most challenges and why?**

    In the Right/Left Wall Tracking, the algorithm maintains a specified distance from the walls. The advantage of this is that for straight corridors, the algorithm is able to effectively track one side of the wall, due to the symmetry of the environment. However, when the environment curves or unexpected obstacles occur, this mode of tracking fails as only one wall is being tracked. Specifically, when there is a left curving corridor in Right Wall Tracking, the AEV maintains a constant distance with the right wall, which causes a collision. This situation also occurs for a right curving corridor in Left Wall Tracking.

    In Center Tracking, the algorithm maintains an equal distance between the left and right walls sensed. This performs well in a straight corridor setting. However, when encountering sharp turns and obstacles in front of the vehicle, collisions occur. However, the navigation is a little smoother in comparison to the single wall tracking modes, as both walls are being sensed, which gives better navigation during turns.

    Overall, the above modes have difficulties for obstacles in front of the car and sharp corners with non-uniform walls. Both modes also have excessive ‘zig-zagging’ in the driving, which is a result of the feedback system trying to maintain constant distances from the walls.

12. **Task: Explain the impact of the controller gains k_p and k_d on the response characteristics. How do you choose these values? You can refer to the course lecture slides for discussion on second-order systems response characteristics.**

    k_p and k_d are the gains within the proportional control system with velocity feedback. This system adjusts the steering angle and commands for velocity based on the error feedback sensed. K_p is the proportional gain, which controls the system output by proportionally adjusting the error between the desired input (ex. speed) and the actual output.

    The k_p, which is proportional gain, impacts the response characteristic by controlling how much the error signal is corrected based on the error feedback. Essentially as k_p is larger, the correction also increases, thus the response becomes faster. The downside to this is that overshoot and oscillations also occur as a side product. Thus there needs to be a balance between having a large enough k_p to correct quickly enough to compensate for the error, while also limiting the overshoot and oscillations as per the system's requirements.

-END-