Artificial potential field algorithm.

Mustafa Poonawala
November 4, 2021

1 Summary

This algorithm provides a way to track an existing path generated by another planning algorithm like A* or RRTs, while avoiding obstacles that were unknown earlier at the path planning stage. Thus the artificial potential field algorithm allows us to merge the tracking component and the reactive component of driving a vehicle in a real environment. The artificial potential field algorithm uses the analogy of charged particles and the forces they exert on each other. We assume that the car and the obstacles are similarly charged and thus repel each other, and the goal and sub-goal points are oppositely charged from that of the car and thus attract it towards them. Subsequently, the obstacles' repelling forces and the goals attracting forces result in motion of the car from start to goal while avoiding the obstacles.

2 Repelling potential:

$$v_i = \frac{1}{n(q,O_i)+d_o} + \frac{n(q,O_i)}{(d_l+d_o)^2} \qquad 0 \le n(q,O_i) \le d_l \tag{2.1}$$

$$v_i = \frac{1}{d_l+d_o} + \frac{d_l}{(d_l+d_o)^2} \qquad n(q,O_i) \ge d_l \tag{2.2}$$

Where $n(q,O_i)$ is the non holonomic distance between the car and the O_i obstacle. d_l is the obstacle influence distance. Obstalces further than d_l will not have any forces on the car. d_o is a constant introduced to prevent forces from going infinite when $n(q,O_i)$ becomes 0.

3 Non-holonomic distance:

In equations 2.1 and 2.2 we use the non-holonomic distance because a car is follows non holonomic constraints while moving toward an obstacle thus the Euclidian distance may be smaller than the distance the car would have to travel to reach that obstalce. Thus the non holomic distance is a better representation of the distance of that obstacle from the car. To calculate the non-holonomic distance $n(q, O_i)$ we use the following equations:

$$n(q,O_i) = d(q,O_i) \qquad \alpha_i = 0 \tag{3.1}$$

$$n(q,O_i) = \frac{L}{sin\alpha_i} \times \alpha_i + d(p_L, O_i) \qquad d(q,O_i) > L \tag{3.2}$$

$$n(q,O_i) = \frac{d(q,O_i)}{sin\alpha_i} \times \alpha_i \qquad d(q,O_i) \le L \tag{3.3}$$

Where $d(q, O_i)$ is the euclidian distance between the car and the ith obstacle and α_i is the angle between ith obstacle and the car's x axis. Also, L is the look-ahead distance.

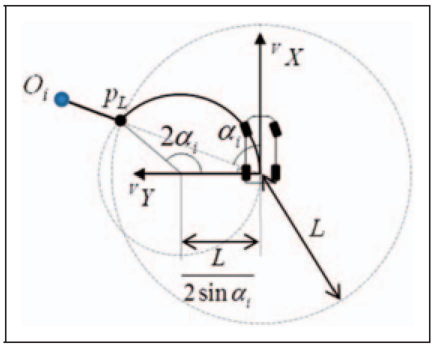

Figure 3.1: Non-Holonomic distance

4 Repelling force:

Now we know, force f_i exerted by the ith obstacle O_i via the potential v_i is given as $f_i = -\bigtriangledown v_i$. Thus we have the force equation as:

$$f_i = -\left(\frac{1}{(d_l + d_o)^2} - \frac{1}{(n(q,O_i) + d_o)^2}\right)\boldsymbol{u_i} \qquad 0 \le n(q,O_i) \le d_l \tag{4.1}$$

$$f_i = 0 \qquad n(q,O_i) \ge d_l \tag{4.2}$$

$$\boldsymbol{u_i} = \frac{{}^V\boldsymbol{P}_{O_i}}{|{}^V\boldsymbol{P}_{O_i}|} \tag{4.3}$$

$\frac{{}^V\boldsymbol{P}_{O_i}}{|{}^V\boldsymbol{P}_{O_i}|}$ indicates the direction vector from the car to the ith obstacle in the car frame. Thus each force f_i is the repelling force directed from ith obstacle to the car.

The total force exerted by all the obstacles is the resultant of all the individual forces. $\boldsymbol{F} = \sum_{i=1}^{N} f_i$

We calculate the resultant force and it's orientation by taking the components of force along the car's x and y axes and summing them up.

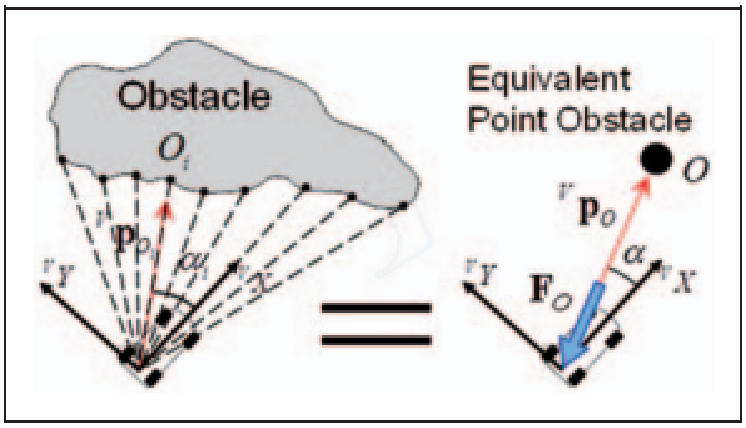

Figure 4.1: Resultant obstacle point.

$$F_x = \sum_{i=1}^{N} f_{x_i} \tag{4.4}$$

$$F_y = \sum_{i=1}^{N} f_{y_i} \tag{4.5}$$

$$\alpha = atan\left(\frac{F_y}{F_x}\right) \tag{4.6}$$

Where α is the angle between the resultant force and the car's x axis. Since the car is a non-holonomic robot we cannot steer or move directly in the "anti-direction" of the repulsive force, so we steer away from the direction of the resultant force and calculate the radius of steering away as follows:

$$R_{avoid} = \frac{-sign(\alpha)}{K_a|F|} \qquad \alpha \neq 0 \tag{4.7}$$

$$R_{avoid} = \frac{1}{K_a|F|} \qquad \alpha = 0 \tag{4.8}$$

where K_a is a tuning parameter. Higher K_a means lower radius of curvature of the path to avoid the obstacle and thus a sharper turn. Therefore K_a should be increased to avoid obstacles more aggressively.

5 Attractive force:

To follow the path we use a pure pursuit controller with the well known equation to track the next look ahead point as below:

$$R_{track} = \frac{L^2}{2y_L} \tag{5.1}$$

Where L is the look-ahead distance and y_L is the y component of the look-ahead point in the car's frame. It should be noted that R_{track} is not just the magnitude of the radius of the trajectory to reach the next look-ahead point. It's sign means the direction in which the car needs to steer. For our case negative R_{track} corresponds to turning right and positive to turning left.

6 Resultant steering:

Now that we have the R_track and R_{avoid} we, just combine the 2 to get the resultant direction that the car should steer towards as follows:

$$\kappa = \frac{1}{R} \tag{6.1}$$

$$\kappa_{total} = \kappa_{track} + \kappa_{avoid} \tag{6.2}$$

$$\delta = b \times \kappa_{total} \tag{6.3}$$

δ is the steering angle and b is the wheelbase. The wheelbase term can just be interpreted as a proportional term.

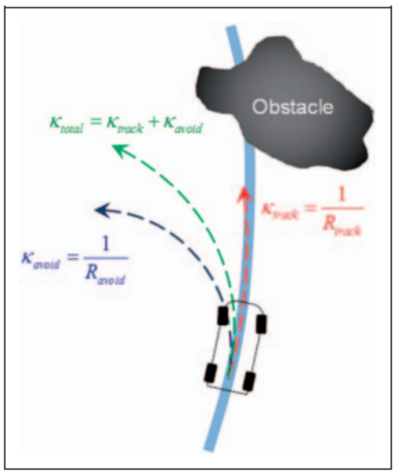
Obstacle
$\kappa_{total} = \kappa_{track} + \kappa_{avoid}$
$\kappa_{track} = \frac{1}{R_{track}}$
$\kappa_{avoid} = \frac{1}{R_{avoid}}$

Figure 6.1: Resultant steering

7 Velocity planning:

Thus we have the steering angle at each iteration based on next look-ahead point and the obstacles around. For velocity planning we use a simple equation as follows:

$$v = v_{max} - K_F \times |F| - K_{delta} \times |\delta| \tag{7.1}$$

Where v_{max} is the maximum velocity and K_a and K_δ are tuning parameters. The effect of planning velocity through this equation is that we slow down where there are a lot of obstacles around and when the steering angles are large.

8 IMPLEMENTATION:

We use 3 subscribers in the node. First is the subscriber to the "/path" topic which collects the path generated by the path planner like Dijkstra, RRT and saves it as a class attribute. The second subscribes to the "/scan" topic and the third to the "/pose" topic. The pose callback uses the car's current pose and the path and calculates the best look-ahead point to track at that instant and calculates the steering angle κ_{track}. The scan callback uses the latest laser scan and calculates the resultant repulsive force based on the scan ranges and thus calculates κ_{avoid}. It then adds it to κ_{track} to get κ_{total} and calculates the velocity v as per equation (1.16) and then publishes the steering angle and velocity to the drive command topic.

9 TUNING:

We will split the tuning task in 2, tuning of the lateral motion and longitudinal motion.
Lateral tuning: Firstly, let us recollect the tuning parameters that we have and the effect they have on the behaviour of the lateral motion:

- d_l: Obstacle influence range.(refer eq 2.1,2.2,4.1) This allows us to look at obstacles only within a certain range. Based on the environment and the speeds we are aiming for, we do not need to look at obstacles that are too far. In our lab setting we set this to 1.5 meters as we are going significantly slow and looking at obstacles beyond that is not productive.

- K_a: As can be seen from equations 4.7 and 4.8, K_a helps us control the radius of curvature of the path that the car takes to avoid the resultant obstacle. And radius of curvature is inversely proportional to the steering angle of the car. A higher K_a results in smaller R_{avoid} and thus bigger steering angle command. To summarize:
  Higher K_a = More aggressive obstacle avoidance

Tuning Methodology: Now, the idea behind tuning the value of K_a is to pick a value that is sufficiently large to avoid obstacles but not too large because then obstacle avoidance would dominate the car would drift away from the path completely. Based on separate tuning of the pure pursuit controller we have chosen a Look ahead distance 'L' of 0.9 meters. This allows us to track the path smoothly. According to equation 5.1 let's see what values of steering command we can expect from the pure pursuit controller to track the next look ahead point.The case where we would expect the strongest steering command is when the look ahead point is exactly on the y-axis of the car as shown in the image 9.1. And as per 5.1 the steering angle would be $\frac{2L}{L^2}$ that is $\frac{2}{L}$ that is $\frac{2}{0.9}$, which comes to 127.32 degrees.[1] So in the worst case the tracking

[1]NOTE: Obviously a steering angle of 127.32 degrees is not possible but we use them as it creates a competing behaviour between the steering angle generated by the obstacle avoidance component of the mechanism. Recall from eq:6.2 that the steering angles by tracking and avoidance components add up(compete).If we to just clip the angles before they get a chance to compete we would just get a maximum steering angle in either direction or a zero. Taking an example suppose K_{track} is 132(positive steering is left) so it becomes 17 and K_{avoid} is -200 which becomes -17 and so the sum would be 0 meaning the car would go straight. This is problematic since -200 is much more than +132 and the car should have gone to the right to avoid the obstacle.

component will demand a steering angle of 127.32 degrees. This is a very rare scenario and upon testing we have found that the K_{track} is generally around 40-50 degrees in the course of tracking a path. So we need to set K_a that generates somewhat similar steering angles so that there is a healthy competition between the tracking and avoidance components.
To come up with K_a we observed the magnitude of the repulsive force generated by the equations in our lab setup and saw it is around 200-300N while approaching obstacles and higher when getting extremely close to obstacles. So a K_a that gives 40-50 degrees of steering angles would be sufficient while approaching the obstacles referring to the plot 9.2 below we found 0.003 is a good value.

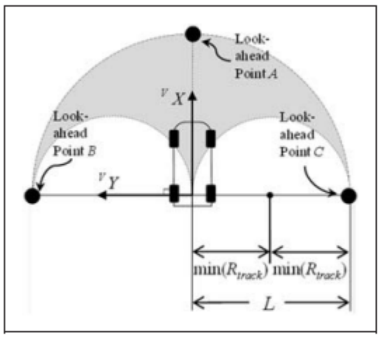

Figure 9.1: Tightest steering situation

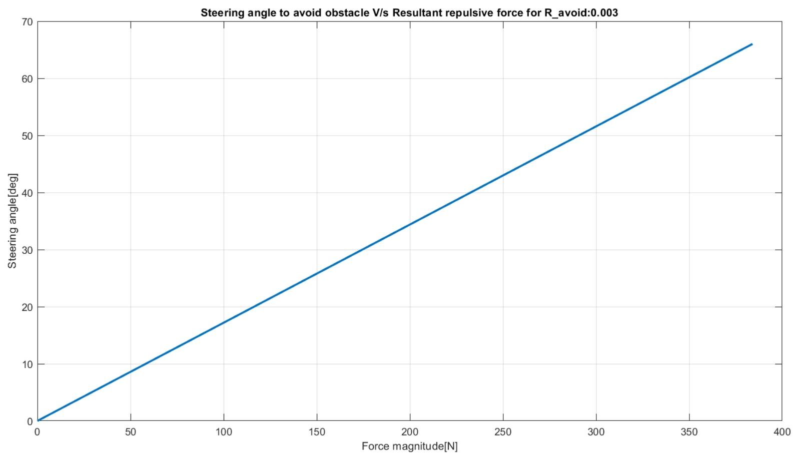

Figure 9.2: Resultant steering

Issues and solution: This method of tuning works for the most part but there is one issue. When there is an obstacle in the way of the path, the car is able to steer away from it but it

If we don't clip them before adding we get 132-200 = -68 which becomes -17 and the car steers to the right as it should.

turns back into the path a bit early, this causes the rear end of the car to brush against the obstacles. It can be seen in this video. Link to video. The reason for this is that the LIDAR is in the front of the car, and as soon as the LIDAR crosses the obstacle it cannot sense the obstacles behind it and hence thinks it is clear to steer back to the path. The first method that we tried to resolve this issue was to increase K_a so that the car keeps a bigger distance from the obstacles. This way, even when the car turns into the path as soon as the LIDAR clears the obstacle there is enough space between the car and the obstacle to make the turn without collision.
However, increasing K_a to fix this issue creates another issue. With high K_a the avoidance component is quite strong and even small forces cause large steering angles. So even if α from eq 4.7 is small we get a large steering angle. Say for example, the resultant obstacle is 15degrees to the right of the car, yet we get a steering angle which is very high towards the left. Since the car turns to the left so strongly, at the next instant the car steers strongly to the right. This causes oscillating motion of the car. Video of this behaviour can be found here. Link to video Link. To fix this, we use an increasing the value of K_a as $|\alpha|$ increases. we use this equation to govern the selection of K_a

$$K_a = 0.003 + side\_safety\_factor \times |\alpha^5| \tag{9.1}$$

This way for when the resultant obstacle is somewhat in front of the car we use the nominal K_a and as the resultant obstacle goes to the side of the car we increase K_a. Note that 'resultant obstacles to the side of the car' represents the scenario in which we are passing by an obstacle closely. At this moment we increase K_a. The plot below shows this. The side safety factor is set as 8×10^{-13} The video for final performance can be found here.Link to video

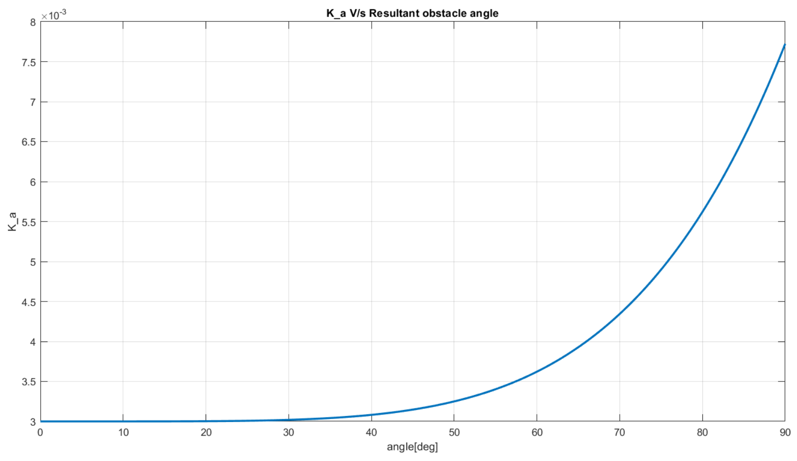

Figure 9.3: Obstacles to the side generate higher K_a values

Longitudinal Tuning: For longitudinal tuning we run the car around the track with a constant speed and observe the force and steering commands given. Based on these values we choose a v_{max}, K_F and K_δ. Due to the small size of the lab setup, we choose v_{max} as 0.8m/sec.
Future scope in tuning: We can run the car in a larger environment and tune it to run at higher speeds and observe if the algorithm is able to perform sufficiently well at higher speeds.