Artificial potential field algorithm.

Mustafa Poonawala
October 30, 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. We use the analogy of charged particles and potential fields and the forces they exert on charged particles. 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.

1. Repelling potential:

$$v_i = \frac{1}{n(q,O_i) + d_o} + \frac{n(q,O_i)}{(d_l + d_o)^2} \qquad 0 \le n(q,O_i) \le d_l \tag{1.1}$$

$$v_i = \frac{1}{d_l + d_o} + \frac{d_l}{(d_l + d_o)^2} \qquad n(q,O_i) \ge d_l \tag{1.2}$$

   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.

2. To calculate the non-holonomic distance $n(q,O_i)$ we use the following equations:

$$n(q,O_i) = d(q,O_i) \qquad \alpha_i = 0 \tag{1.3}$$

$$n(q,O_i) = \frac{L}{sin\alpha_i} \times \alpha_i + d(p_L,O_i) \qquad d(q,O_i) > L \tag{1.4}$$

$$n(q,O_i) = \frac{d(q,O_i)}{sin\alpha_i} \times \alpha_i \qquad d(q,O_i) \le L \tag{1.5}$$

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.

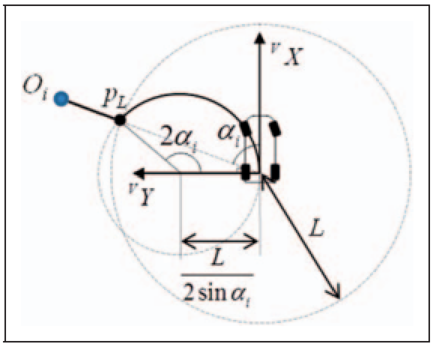

Figure 1.1: Non-Holonomic distance

3. 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:

$$f_i = -\left(\frac{1}{(d_l + d_o)^2} - \frac{1}{(n(q, O_i) + d_o)^2}\right)\boldsymbol{u_i} \qquad 0 \le n(q, O_i) \le d_l \tag{1.6}$$

$$f_i = 0 \qquad n(q, O_i) \ge d_l \tag{1.7}$$

$$\boldsymbol{u_i} = \frac{{}^V\boldsymbol{P}_{O_i}}{|{}^V\boldsymbol{P}_{O_i}|} \tag{1.8}$$

   $\frac{{}^V\boldsymbol{P}_{O_i}}{|{}^V\boldsymbol{P}_{O_i}|}$ 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.

$$F_x = \sum_{i=1}^{N} f_{x_i} \tag{1.9}$$

$$F_y = \sum_{i=1}^{N} f_{y_i} \tag{1.10}$$

$$\alpha = atan\left(\frac{F_y}{F_x}\right) \tag{1.11}$$

   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

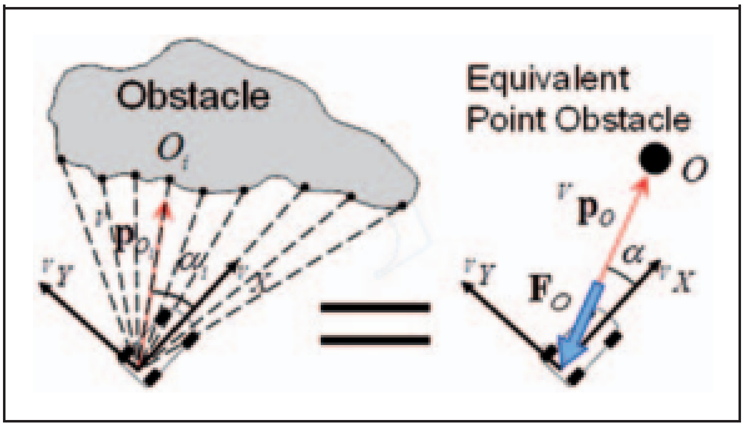

Figure 1.2: Resultant obstacle point.

repulsive force, so we steer away from the direction of the resultant force and calculate the radius of steering away as follows:

$$R_{avoid} = \frac{-sign(\alpha)}{K_a|F|} \qquad \alpha \neq 0 \tag{1.12}$$

$$R_{avoid} = \frac{1}{K_a|F|} \qquad \alpha = 0 \tag{1.13}$$

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.

4. 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:

$$R_{track} = \frac{L^2}{2y_L} \tag{1.14}$$

$$\tag{1.15}$$

   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.

5. 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:

$$\kappa = \frac{1}{R} \tag{1.16}$$

$$\kappa_{total} = \kappa_{track} + \kappa_{avoid} \tag{1.17}$$

$$\delta = b \times \kappa_{total} \tag{1.18}$$

δ is the steering angle and b is the wheelbase.

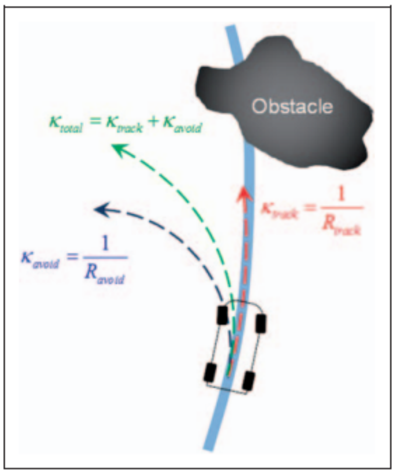

Figure 1.3: Resultant steering

6. 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:

$$v = v_{max} - K_F \times |F| - K_{delta} \times |\delta| \tag{1.19}$$

   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.

7. 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.