Potential Function Based Motion Planning

SC627 Assignment 2, Dhananjay Tiwari

March 5, 2022

In this report an implementation of potential function based motion planning algorithm is discussed for a mobile the robot. The robot moves in a 2D plane and has three degrees of freedom - translation in x and y directions, yaw angle about z-axis. The objective is to design an iterative scheme for the motion update of the robot so that it is able to reach a goal point in the $x-y$ plane starting at some other point. The arena also has some obstacles in the form of polygons that the robot is suppose to avoid while moving. The potential function based approach contructs a potential field in the arena with the attractive potential energy associated with the goal point and repulsive potential energy associated with the obstacles. To attain the state of minmum potential energy, the robot moves in the direction of decreasing potential energy or decreasing potential gradient. In the current implementation, the potential functions are given by,

$$U_{att}(q) = \begin{cases} \frac{1}{2}d^2(q, q_{goal}), & \text{for } d(q, q_{goal}) \leq d^*_{goal} \\ d^*_{goal}\chi d(q, q_{goal}) - \frac{1}{2}\chi(d^*_{goal})^2, & \text{otherwise} \end{cases} \tag{1}$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{d_i(q)} - \frac{1}{Q^*_i})^2, & \text{for } d_i(q) \leq Q^*_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In the above, the function $Q \times Q \ni (q_1, q_2) \rightarrow d(q_1, q_2) \in \mathbb{R}$ is a distance metric giving the distance between the two points on the configuration space. Similarily the function $Q \ni q \rightarrow d_i(q) \in \mathbb{R}$ gives the distance of a point $q \in Q$ from the i^{th} obstacle. The terms $d^*_{goal}, \chi, \eta, Q^*_i$ are constants. The above functions show that the attractive potential achieves a minima (= 0) at q_{goal} and as we move away from the goal, the function increases first quadratically and then linearly with respect to the distance. Further the gradient of the attractive potential remains continuous. On the other hand, if we notice the repulsive potential, near the obstacles it attains extremely high values and as we move away the potential falls. A plot of the potential function is shown for $d^*_{goal} = 2, \chi = 0.8, \eta = 0.8, Q^*_i = 2$ in the figure 2 for the obstacles shown in the figure 1. The q_{goal} is chosen to be $(5, 3)$.

We use the following iterative scheme to update the position of the robot

$$q_{k+1} = q_k - \alpha \nabla U(q) \mid_{q=q_k}, \eta > 0 \tag{3}$$

where $U = U_{att} + U_{rep}$. The above iterative scheme is used for motion planning of the robot, with $\alpha = 0.1$, starting point at $(0, 0)$. The path followed by the robot is shown in the figure 3. The figure shows two plots, one in green line and other one in the red-dotted line. The green plot is obtained using 3 and the red plot is obtained by solving the following dynamics

$$\dot{q} = -\nabla U(q) \tag{4}$$

using Runge-Kutta (order 4) solver (It is also observed even if we use Euler Forward scheme, which is implemented in *potential_function_planar_script.py* in the submission). While the green

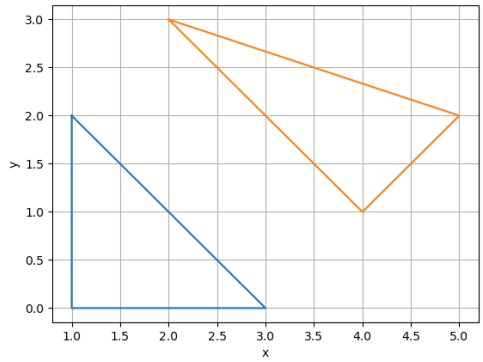

Figure 1: Obstacles

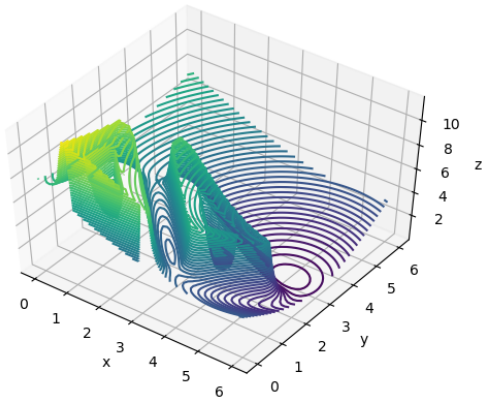

Figure 2: Potential Function (The value is chopped to a finite number near the obstacles)

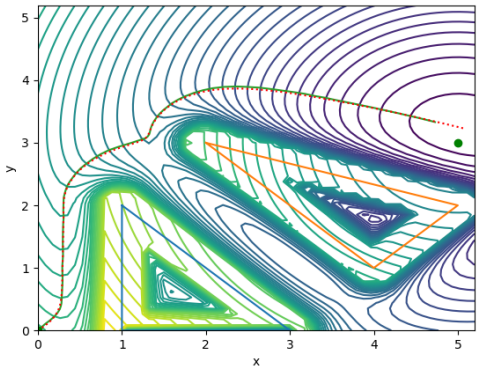

Figure 3: Trajectory of the robot

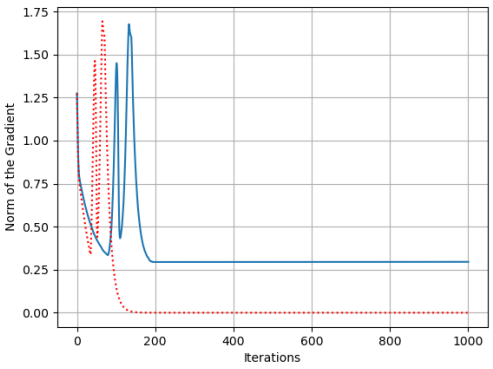

Figure 4: Potential gradient along the path of the robot

line follows red line, the lines do not converge to the goal position $(5, 3)$. The red line does not converge as in the close neighbourhood of the goal position, the gradient becomes very small and hence the driving force for the bot motion becomes small. Further, the green line, which represents the actual trajectory in the ROS simulation stops due to some noise in updating the current position of the robot using *moveTurtle()* function. The current position obtained from *moveTurtle()* becomes constant as the bot reaches near the goal. The figure 4 shows the potential gradient for the corresponding plots. For the red plot, the potential gradient is almost zero and the bot cannot further to reach the goal. In other words, in the close vicinity of the goal the potential function is almost flat. The gradient along the actual path suddenly becomes constant as the bot halts in that position.

Failure of Potential Function Planner

Now the goal position of the bot is changed to $(2, 2)$. The trajectory of the bot is shown in the figure 5 and the corresponding gradients are shown in 6. The robot stops at near point $(0.3, 2)$, and this is because the potential field function is almost flat at this neighbourhood but it does not attain its minimum. The claim is also evident from the plots shown in the figure 6 (The red plot, because the blue has error due to *moveTurtle()*). A careful look at the 3D plots 7, 8, 9 also shows the small flat region around this point. These results show that the potential function has to be chosen carefully so that the sum of attractive and repulsive potential does not make a region other than the goal flat. However, this is a extremely difficult to figure out a function, when the terrains also become extremely complex. Thus, the potential function planner can is suitable for simple terrains, but need not work for complex environments.

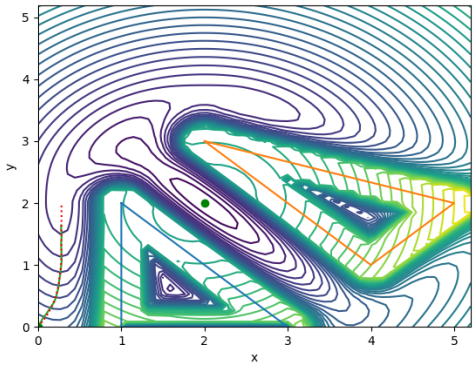

Figure 5: Trajectory of the robot

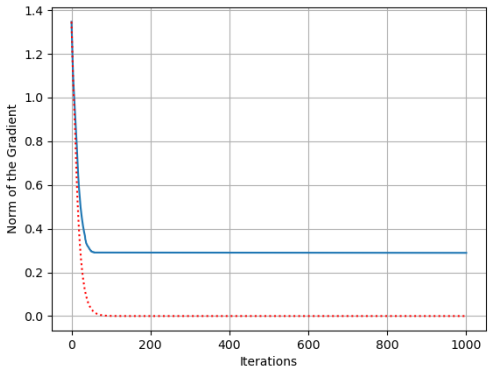

Figure 6: Potential gradient along the path of the robot

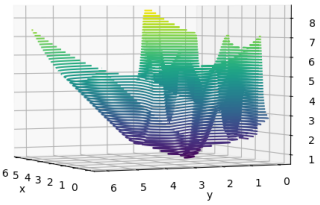

Figure 7: Potential Function, view 1

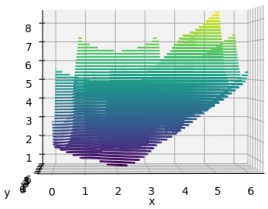

Figure 8: Potential Function, view 2

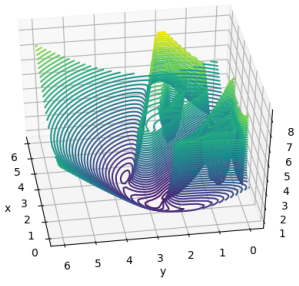

Figure 9: Potential Function, view 3