Capstone Project

Runaway Robot

1, Definition

Project Overview

This project came from an optional final projecy in Artificial Intelligence for Robotics. This project assumes a situation like: A robotics company named Trax has created a line of small self-driving robots designed to autonomously traverse desert environments in search of undiscovered water deposits. In order to maneuver itself, a Traxbot can do one of two things: it can drive in a straight line or it can turn. So to make a right turn, A Traxbot will drive forward, stop, turn 90 degrees, and then continue driving straight. All of sudden, this bot has gotten lost somewhere in the desert and is now stuck driving in an almost-circle: it has been repeatedly driving forward by some step size, stopping, turning a certain amount, and repeating this process... Luckily, the Traxbot is still sending all of its sensor data back to headquarters.

Therefore, we need to track the lost target. More specifically, the project programes a robot to track a target by using some methods in robotics like particle filter[1] and PID control[2]. Particle filter helps the robot to speculate the next location of the target, and PID control enables the robot to move more smoothly.

This project is from the optional final project in Udacity's Artificial Intelligence for Robotics class. There is no data set, but basic classes, like robot class or matrix class are defined in the class.

---

[1] http://robots.stanford.edu/papers/thrun.pf-in-robotics-uai02.pdf

[2] http://umpir.ump.edu.my/415/1/Muhammad_Bin_Mazlan_3182.pdf

Problem Statement-

Since the problem I assume is the broken target, which moves in cricle, in a desert, simplified world would be no-bound virtual 2d and the target which moves in circle. This project aims to programe a robot to track a target, which moves in circle with some noise. All the information the robot can get from a target is sensing data that where the target is. However, this information is not necessarily equal to the exact location because of the noise. Both the robot and the target move in the same speed. This makes the problem a little bit complicated. Three steps are needed to solve this problem. First, the robot will track the target based on the sensing information. Second, the robot intends to move in smaller circle than the target because the robot never catches up with the target if it moves in the same circle with the target's. Third, when the distance between the target and the robot become less than the distance one can afford in one step, the robot will try to catch the target by estimating the target's next position. In this project, the expected result is that the robot can catch the target in less than 1,000 steps.

1: The robot try to pursue the tareget by using present sensing data, means that not estimating the next location for the target. In addition, it helps the model avoid predicting poorly since the particle filter does not work well with the lack of data.

2: After 20 steps, the target starts estimatig the next location by using the particle filter, and controlling its motion with PID control so that it would move in slightly small circle.

3: Once the distance between the target and the robot become smaller than the distance they are allowed to move, the robot try to cath the target, means that no PID control, only by using the particle filter.

4: If the target could not make it, the process 2 and 3 would be repeated.

Metrics

The metric to evaluate my robot is based on the number of actions it took to find the target. 50 trials would be appropriate since cost of culculation is not expensive so that I would improve scores many times. The equation for the evaluation is given by sum of steps for each trial devided by total number of trials. If one trial fails to track the target, it is taken as 1,000 steps.

$$1/n \sum steps$$

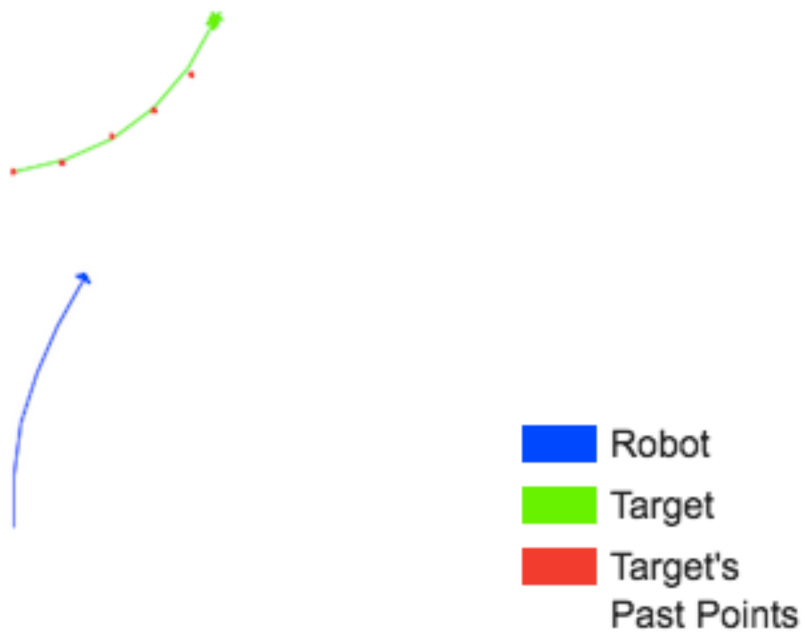
Where ‘n’ is the numebr of the trials and ‘steps’ are the number of steps to the goal for each trial.

2, Analysis

Environment Specifications

Environment

This project assumes 2 dimensions virtual environment to model desert. This environment is expressed as (x, y) with continuous values and does not have any obstacles including walls.



Target Specifications

The target is the one object the robot tries to catch. It basically moves in circle and every step the target moves, the robot can obtain sensing information like (x, y) from the target. Basic information like, initial location and how the target moves are given when the target is made. In other words, the target gets the information like location, heading, turning and distance. Random function sets them except for distance, and distance is consistently set in 1.5.

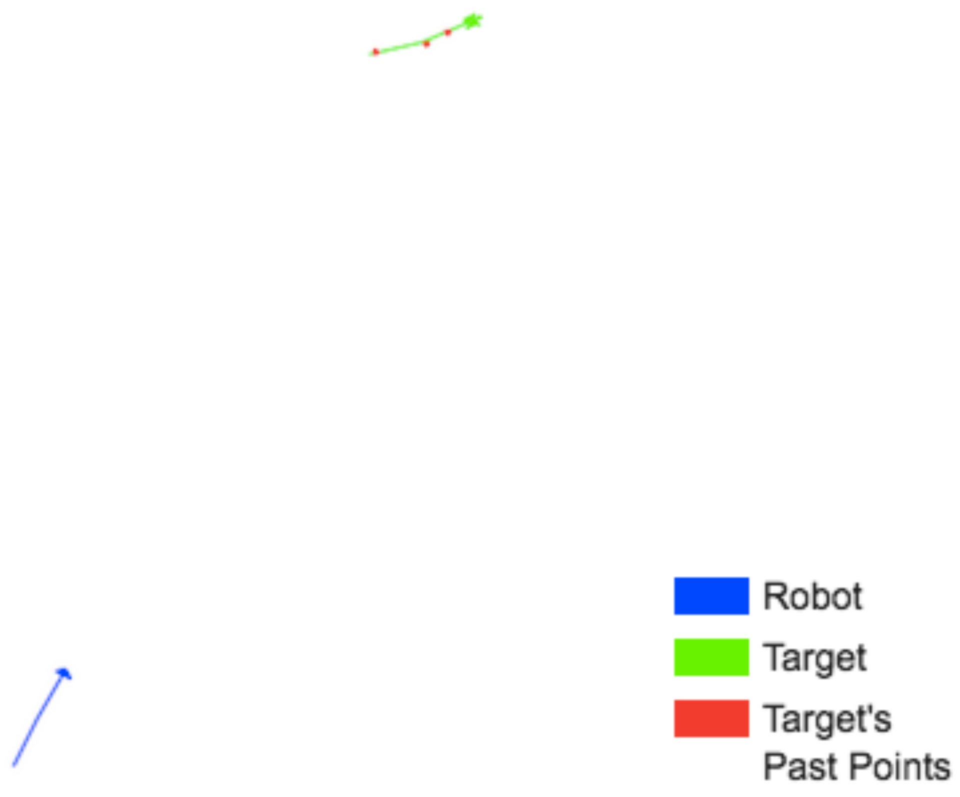
Location: denoted as (x, y), x and y ranging from -10 to 10
Heading: denoted as heading ranging from 0 to 1
Turning: denoted as turning ranging from -2pi/30 to 2pi/30
Distance: cinsistently 1.5

Robot Specifications

In the beginning of each trial, the target is given to random paramators in x, y, heading and turning: x and y are from -10.0 to 10.0, heading is from 0 to 2.0*pi and turning is from -2.0*pi/30.0 to 2.0*pi/30.0. All of them are including decimal points. For example, (x, y, heading, turning) = (0.0, 10.0, 0.0, 2.0*pi/20).

Location: both x and y initially set to 0, once it starts moving, the paramator is controllable
Heading: initially set to 0, once it starts moving, the paramator is controllable
Turning: initially set to 0, once it starts moving, the paramator is controllable
Distance: cinsistently 1.5

Using the same robot class with the target makes the robo. This means that the robot has the same information like location, heading turning anf distance. Unlike the target, all the factors except for distance can be coordinated by any functions, but as the same with the target, distance is the only consistent factor set to 1.5.



The robot’s initial location was (x, y) = (-10, -10) and the target’s was (x, y)=(10,10). The initial target’s (heading, turning) is (0.0, 2pi/30). Both agetnt’s distance paramator is 1.5 consistently

Algorithms and Techniques

Localization

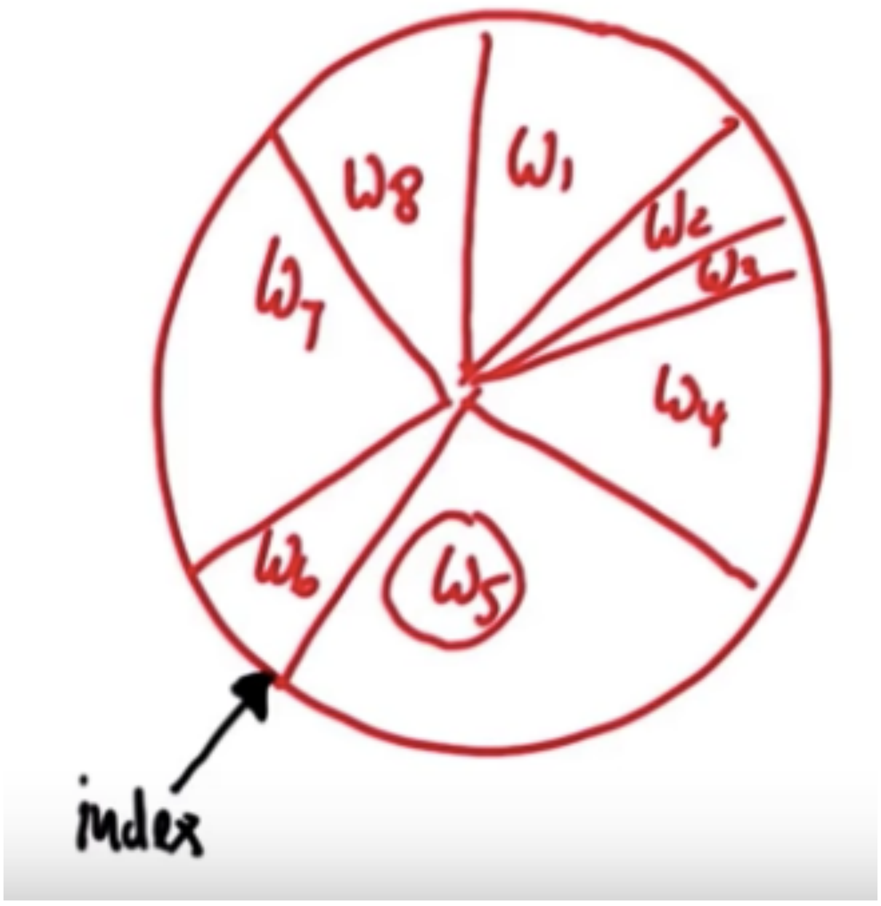
In order to catch the target, the robot needs to estimate a location for the target. The model used a method called localization, which enables us to speculate the location of an object. One of localization techniques is used for speculating the location: it is particle filters, more specifically Monte Carlo Localization[3]. In my model, 500 particles 8 landmarks are set in virtual environment. Particles measure the heading and distance between them and landmarks in every step. Landmarks are chosen by randomly, and they would not move, play a role as a landmark. The distance between landmarks and the target is measured in every run. In other words, the model comapares them with the target's heading and distance in every run. If the distance and heading show some similarity, the particle more likely to get survived, which means that the particle is given some weight by below equation.

$$\prod p\left(measurement\right)$$

Where p(measurement) is the probability of the particle's location in Gaussian distribution given by the distance between the target and the particle as mean and the noise of the sensing as standard deviation, and this is iterated by the numebr of landmarks.

Next step, resampling process need to be done depending on each weight(probability). First, the weight has to be normalized divided by the sum of weights. Resampling, more more specifically sampling wheel, is used in this normalized weights. In the below picture we iterate the circle many times, and every time we pick the one weight. That is to say, the larger space the weight has, the more likely to survive. Through this process, the resampling is conducted.

[3] http://umpir.ump.edu.my/415/1/Muhammad_Bin_Mazlan_3182.pdf



By repeating the above weighting and resampling, the particles could speculate the precise location of the target.

Particle filter has many advantages. It can be used for continnuous values and multimodal distribution. Furthermore, this method is really efficient in low dimensions. For the above reasons, I chose particle filter for localization.

Control

The key point in this project to solve the problem is that the robot needs to move in smaller circle than the target's since they move at the same spped. However, without controlling the robot, the robot would not move smoothly along the small circle. PID control makes it possible for the robot to move as it is expected. PID control is short for Proportional, Integral and Derivative control. My model applys PID control to Turning the robot takes. An equation shown below represents the PID control: Kp, Ki and Kd are paramators and e(t) is a heading difference between the taeget and the robot in my model. This equation means that Turning of the robot becomes slightly smooth, and this would make a huge diffrence in robot's motion.

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\, d\tau + K_\mathrm{d} \frac{de(t)}{dt}$$

Benchmark

The benchmark is made by a naive approach, which always direct the robot in the direction without using PID control. The equation of 'Success trial rate' is given by the number of success trials devidied by the number of trials. The equation of Number of Steps to Catch in Average for a benchmark is given by sum of steps for each trial devided by total number of trials. Also, the stardard deviation for the steps for each is calculated.

Number of trials: 50
Number of success trials:17

$$STA = s/n$$

Where 's' is the numer of success trials and 'n' is the numberof trials.
Success trial rate: 35%

$$NSCA = steps/n$$

Where ‘steps’ is the number of steps to take to the goal in one trial and ‘n’ is the numebr of trials.

Number of Steps to Catch in Average(NSCA): 696.18

Stardard deviation: 278.95

3, Methodology

Preprocess

Since this project does not need preprocessed data before the program starts, there is no need to preprocess data.

Implementation

First 20 trials, the robot just try to chase the past location of the target to minimize the distance between the robot and the target since particle filter needs some trials to start estimatig location more closely. Originally, particle filter is for knowing where the object is, but it needs to estimate next location. In this project, the target’s speed and turning is stable. In addition, the robot can get the sensing data from the target. By using these information, the model enables the robot to estimate next location by particle filter.

After 20 trials, the robot starts moving in smaller circle than the target’s by using both particle filter and PID control. By introducing PID control, the robot changes its heading slightly. They enables the robot to move along the target. Once the target is close enough to catch the target in next step, it trys to catch the target without using PID control since the control cahges the heading sloghtly.

Refinement

Paramator tuning for PID control was done by twiddle function used in Artificilal Intelligence for Robotics[4]. In the twiddle function, the error rate, which is defined as the difference between the actual target’slocation and the estimated location, is calculated. Every time the twiddle is called, the paramator is changed a little bit. And the error rate is calculated and is compared with previous error rate. When the error rate is improved, the function tweak the paramator to the direction as the paramator previously tweaked. If it were not improved, the function would tweak the paramator to the opposite way as the function did previously. By running the twiddle function, the paramators could be optimised.

I implemented this twiddle function by setting fixed random paramators, which enables the model to be generalized. As a redult, Number of Steps to Catch in Average is improved from 329.68 to 289.55. And Success Trial Rate becomes from 88% to 90%.

[4] http://robots.stanford.edu/papers/thrun.pf-in-robotics-uai02.pdf

4, Result

Model evaluation

The model uses particle filter and PID control, while the bencmark model only uses particle filter. And paramator tuning is done by the twiddle function.

Number of trials: 50
Number of success trials:45
Success Trial Rate(STA): 90%
Number of Steps to Catch in Average: 256.48
Standard deviation: 224.54

If the robot fails to catch the target wihitn 1,000 steps, it counted as 1,000 steps for one trial.

Compared with the benchmark result, ‘Success trial rate’ become 2.4 times improved and the reduction of 440 steps is achieved in “Number of steps to catch in average”.

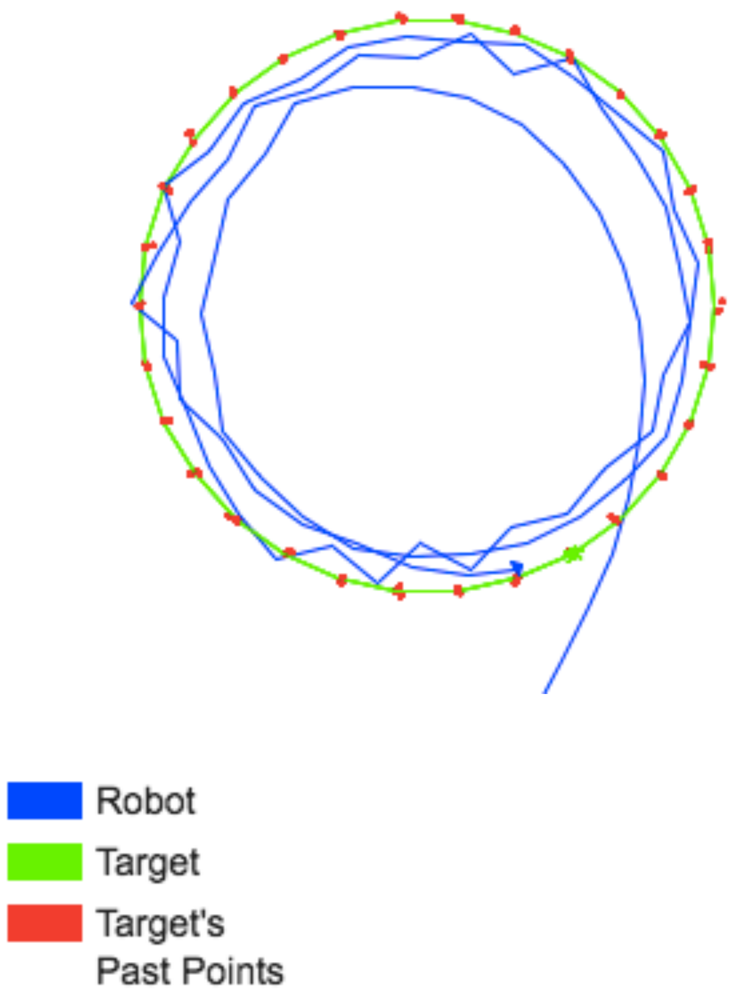
Paramator tuning in PID control by using the twiddle function did improve my redult.

Compared with benchmark’s standard deviation, the result is improved from 278.95 to 224.54 and the model becomes more robustness.

5, conclusion

Visualization

This visualization shows how the robot track the the target. After 20 steps, the robot starts moving in small circle and the robot is close enough to catch the target, it tries to cathch the target.   You can get this buy runnnig visual.py.



Reflection

In this project, I programmed the robot to track the target. First thing I did is to estimate next location for the target by using particle filter. However, just estimating next location for the target is not enough for the robot to catch the target. That is why, I implemented PID control to control the robot to move in smaller circle. Using this method enable the robot to track the target more efficiently as it is suggested in model evaluation.

The interesting point I learned is that PID control makes huge difference in result. First time I learned control method in robotics, I was skeptic about how the method works, but I realized that how important the method is.

Improvement

Other improvements could be kalman filter[5] for localization. And the other control method for control, which uses collective feedback for cotrol[6].

---

[5] http://scholar.google.co.jp/scholar_url?url=https://trac.v2.nl/export/7478/andres/Documentation/INS%2520Kalman/adaptive%2520extended%2520kalman%2520filter%2520for%2520the%2520localization%2520of%2520mobile%2520robots.pdf&hl=ja&sa=X&scisig=AAGBfm33DlN7SDd9rFYmJFY_60sREh3c_A&nossl=1&oi=scholarr&ved=0ahUKEwjB_qSI973QAhVEVbwKHVo8A70QgAMIGigAMAA

[6] http://www.cs.cmu.edu/~bargall/argall-dissertation.pdf

References

"Artificial Intelligence for Robotics" Udacity
https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373

Particle Filters in Robotics, Sebastian Thrun,
http://robots.stanford.edu/papers/thrun.pf-in-robotics-uai02.pdf

Path Follower Mobile Robot Using PID Controller, Muhammad Bin Malzan,
http://umpir.ump.edu.my/415/1/Muhammad_Bin_Mazlan_3182.pdf

Development and Experimental Validation of an Adaptive Extended Kalman Filter for the Localization of Mobile Robots, Leopoldo Jetto,
https://trac.v2.nl/export/7400/andres/Documentation/INS%20Kalman/adaptive%20extended%20kalman%20filter%20for%20the%20localization%20of%20mobile%20robots.pdf

Learning Mobile Robot Motion Control From Demonstration And Corrective Feedback, Brenna D. Argall, http://www.cs.cmu.edu/~bargall/argall-dissertation.pdf