FLYTBASE ASSIGNMENT

Submission By: Yazid Marzuk
Date: July 19th 2023

Goal 1:

The objective of the goal was to make the turtle reach a point specified by the user.

The turtlesim_node interface is divided into cartesian coordinates. (0,0) bottom left and (11,11) top right corner.

To achieve the motion of the turtle from one point to another, we have defined constants of P(kp), I(ki) and D(kd).

The errors are calculated using a feedback loop.

Using the errors and the PID Values, the velocity is calculated and pubished to '/turtle1/cmd_vel' topic.

Goal 2:

The Goal requires us to make the turtle move in a grid like formation. So we define the corner points of the grid that the turtle needs to travel. Then make the turtle travel to each of the grid points using the velocity calculation algorithm used in Goal 1. With addition to that the acceleration and deceleration is limited using another function. The Velocity is increased in steps. And each step is checked with time so that it does not increase more than the specified acceleration. Thus acceleration control is achieved

Grid Formation:

1. Go to starting position as fast as possible.
2. Grid Points are declared which is travelled to by the turtle one after the other.
3. The Turtle Orientation is changed after it reaches a location by the specified angle.

4. The grid points declared specify the x, y and angle of the turtle.
5. The turtle comes to the initial point after the last grid point and restarts the process.

Goal 3:
The radius and velocity of the turtle is given.
The velocity of the turtle is increased in steps like in Goal2.
rt_real_pose is publised with the actual pose of the robot every 5 sec.
rt_noisy_pose is published with actual pose added with guassian noise generated using the random module of python.

Goal 4:
Launch the `turtlesim_node` with one turtle to begin the chase.

1. Use the `circle` function from Goal 3 to initiate the circling motion of the 'Robber Turtle'.

2. Launch the `turtlespawn` node with a 10-second delay. After the delay, call the `spawn` service with random values to spawn another turtle named 'turtle2'.

3. Launch the `chase` node. The node subscribes to `rt_real_pose` and waits for 5 seconds before receiving its first goal position.

4. the chase is completes when the turtles are 3 units apart

5. The problem is sometimes the random turtle is generated already within 3 units of the ‘Robber Turtle’ which results in no motion.

6. The speed of the PT determines how far apart they will be when the next `rt_real_pose` comes in.

7. The size of the circle plays a role as a smaller circle will complete the chase quickly if PT is fast enough.

Goal 5:

PT is faster than RT. Which means it will obviously be ahead of RT.

PT should have a velocity half of RT.

So, Here, the velocity of RT is 2, and that of PT is limited at 1

To improve the chase planning, we use a special node called `chase_limit_accel_vel_plan`. This node takes advantage of the fact that the Robber Turtle (RT) moves in circles.

It waits for the first three positions of RT, which define a unique circle.

Then, it uses a planning function to calculate where RT will be next when it sends a signal.

If the Pursuer Turtle (PT) can reach that position within 5 seconds, PT is commanded to go there. If not, we predict

where RT will be in 10 seconds and repeat the process until we find a reachable position.

This way, we can efficiently plan PT's movements to catch RT. As the RT's real positions arrive, we check if they are within the catching range (3 units) and end the chase accordingly.

Goal 6:

The PT is allowed to access RT noisy pose very 5 sec through rt_noisy_pose.

When the Robber Turtle (RT) moves with uncertain pose data due to noise, it becomes harder for the Pursuer Turtle (PT) to catch it.

The amount of uncertainty depends on how much the pose data fluctuates.

If PT is slower than RT and the noise is too high, PT might not catch RT.

However, the circular motion of RT can still help PT get close enough to end the chase.

But too much uncertainty can make PT's movements unstable. Finding the right balance between noise and PT's speed is important for a successful chase.

REFERENCES

1. TurtleBot Spawn Message:

https://docs.ros2.org/foxy/api/turtlesim/srv/Spawn.html

2. TurtleBot to move to a particular location:
https://wiki.ros.org/turtlesim/Tutorials/Go%20to%20Goal#The_Proportional_Controller

3. Random Gaussian Noice Generation:
https://www.geeksforgeeks.org/random-gauss-function-in-python/

4. Turtle Sim Pose Message Structure:

https://docs.ros.org/en/diamondback/api/turtlesim/html/classturtlesim_1_1msg_1_1__Pose_1_1Pose.html