Turtlebot

Jan Johannsen

November 2023

Contents

List of Figures

1 Introduction

In the field of robotics one of the premier introductory robots is the Turtlebot. The Turtlebot is a robot consisting of a body with two wheels attached to it. These wheels can rotate at different speeds and either forward or backward. Based on these parameters the Turtlebot is then able to move. The following will discuss how a Turtlebots movement can be simulated.

2 Math

2.1 Introduction

In the following mathematical concepts necessary for the Turtlebot simulation will be introduced. The simulation will be done in a Cartesian coordinate system with the bots position being determined by a point in the center of the bot and its current direction by a normalized directional vector.

2.2 Translation

For the Turtlebot to move throughout the plane the simulation will make use of translation. Translation is a transformation which moves every point it is applied to on both the x axis and y axis. To transform the original points x, y to the translated points $x1, y1$ the following formula is used. The above is based of chapter 2.3.2 of [1]

$$x1 = x + e \tag{1}$$

$$y1 = y + f \tag{2}$$

With e describing the movement along the x axis and f along the y axis. One example of a translation moving a form F can be seen in Figure 1.

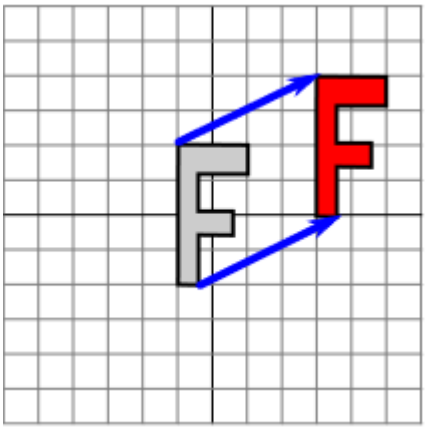

Figure (1) Translation of a form F by (4,2)

2.3 Rotation

While translation can be used to move the bot in a straight line either forward or backwards, the concept of rotation comes into play when the bots wheels move at differing speeds. Rotation in a type of transformation which rotates every point it is applied to around the origin (0,0) by the same angle. When rotating the point (x, y) by r radians around the origins the following point $(x1, y1)$ can be calculated using the following formula. The above is based of chapter 2.3.3 of [1]

$$x1 = cos(r) * x - sin(r) * y \tag{3}$$

$$y1 = sin(r) * x + cos(r) * y \tag{4}$$

An example of rotation being applied to a form F can be seen in the Figure 2.

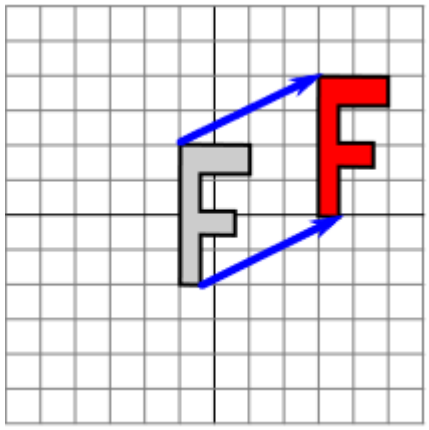

Figure (2) Rotation of a form F by negative 135 degrees

3 Simulation of the Turtlebot

3.1 Problem Description

The following chapter will discuss how to simulate a Turtlebot moving through a flat plane. This plane is represented by a Cartesian Coordinate System with the x and y axis being in centimeters. The simulation will continuously produce new positions for the center of the bot as well as the direction the bot is currently facing in form a normalized directional vector. The inputs for these calculations are w_1 and w_2 and the duration of a time step until the calculation of a new value. The w values describe how often each wheel is turning per second and also their direction based on how the w is signed. To calculate the position of the wheels as well as their values for both the length of the wheel base and the wheel radius have to be specified ahead of the simulation. To simplify the problem acceleration and slowing down of the robot are instantaneous and the forces counteracting this are omitted from the simulation.

3.2 Program Structure

The program, the results are simulated in is written as a python script containing a Turtlebot class. This class contains fields for two vectors to store the current center position and the normal of the bot. In addition to that a current w value is stored for each wheel. The bot instance can be moved through a method move(omega1, omega2, timestep). This method takes new w values for both wheels and a duration of the next step. Internally move splits the movement into different cases. If both wheels rotate in the same direction at the same speed the bot moves straight into either forward or backward direction. If the wheels rotate at the same speed but in different directions the bot spins around its own center. If the w values are both positive but not equal, the rotation point lies outside of the bot. If the wheels rotate at different speeds and are signed differently the rotation point lies inside of the bot on its wheel base, but skewed closer to the slower wheel. Depending on the case a submethod is called to handle the change in position and normal of the bot. To visualize the results of the simulation these results are stored at each time step and plotted onto a 3d graph, with x and y being the bot coordinates and z being the time axis.

3.3 Experiment Description

Taking the implementation described above the bot of the movement can now be simulated. To interpret the results of the movement inside the simulation the bot is given different w values to account for each case discussed above. The exact values are

1. $w_1 = 4$ r/s and $w_2 = 4$ r/s same direction, same speed
2. $w_1 = 4$ r/s and $w_2 = 2$ r/s same direction, different speed
3. $w_1 = 4$ r/s and $w_2 = 0$ r/s one wheel with no rotation
4. $w_1 = 4$ r/s and $w_2 =$ -2 r/s different direction, different speed
5. $w_1 = 4$ r/s and $w_2 =$ -4 r/s different direction, same speed

With w being in rotations per second. The whole experiment is set to simulate 5 seconds of movement with the cases lasting one second each. To further visualize the differences between each case the results. The figures include subsequent simulations with each case being rerun alone for 5 seconds. In addition to this the results are split into two figures. The first one containing the center position for the bot displayed in green and the normals of the bot in blue. The second one contains only the normals of the bot to allow for better scaling of the x and y axis.

3.4 Interpretation of Results

In figure 3 the cases are combined. The bot starts of moving in a straight line from second zero to one. In figure 9 the normal is a straight line from second

zero to one. The following cases result in the bot deviating from its straight path, with the rotation being quicker the closer w values get to $w_1 = -w_2$. The changes in position can be seen in the figures 5, 6, 7, 8. With the exception of figure 8 the bot center rotates around a point. The distance to the middle point of the rotation gets shorter from case 2 through case 4. In figure 8 the bot is rotating around its own center, thus there is no change in position to reflect this. However when looking at the normal in figure 8 its shown that the bot still rotates. Furthermore this rotation is faster than previous cases. This can be determined when comparing the figures 11, 12, 13, 14.

3.5 Result Figures

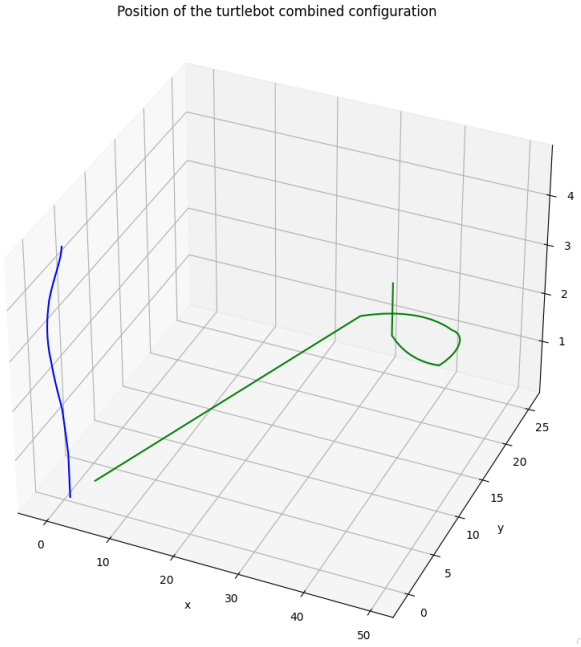

Figure (3) Turtlebot positions with all following configurations

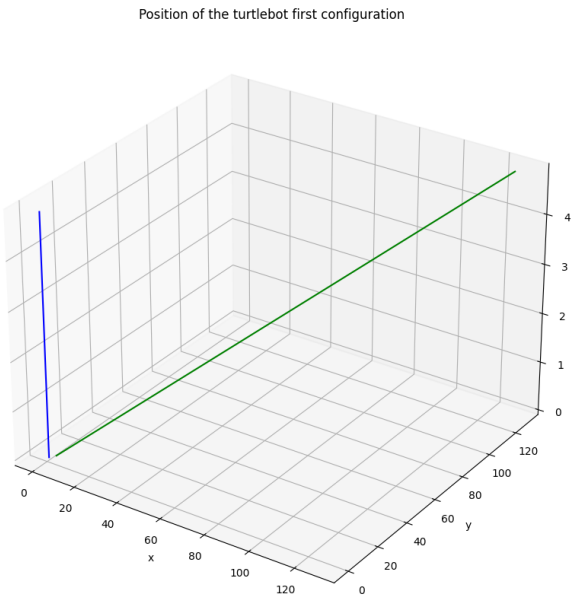

Figure (4) Turtlebot positions with: $w_1 = w_2 = 4$

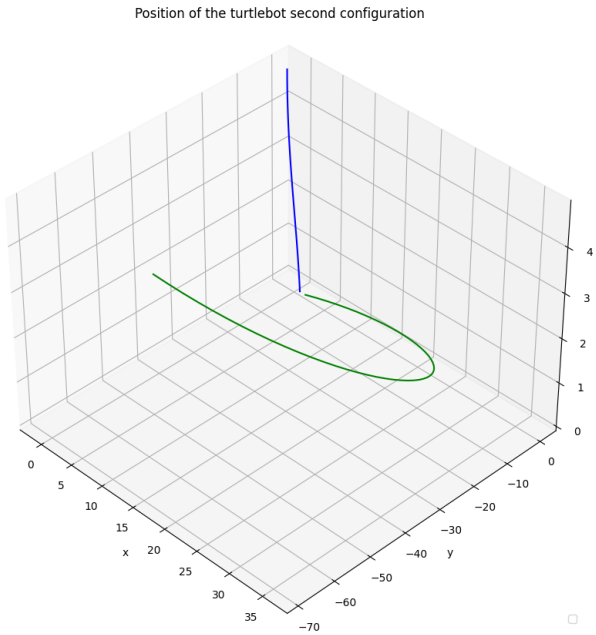

Figure (5) Turtlebot positions with: $w_1 = 4$ and $w_2 = 2$

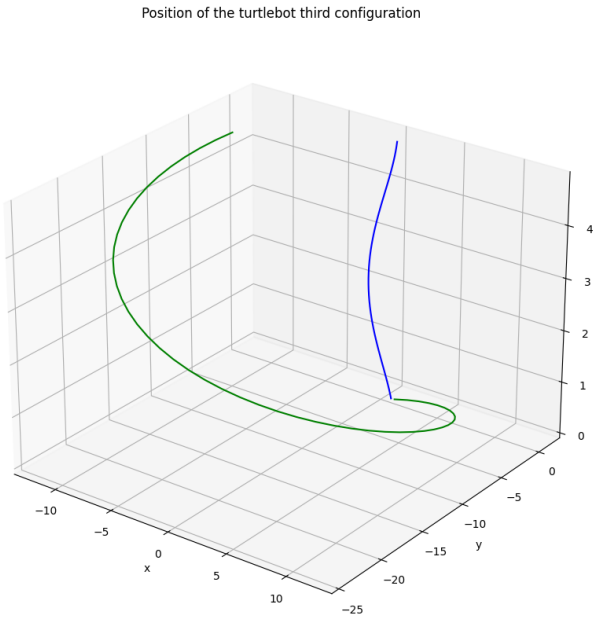

Figure (6) Turtlebot positions with: $w_1 = 4$ and $w_2 = 0$

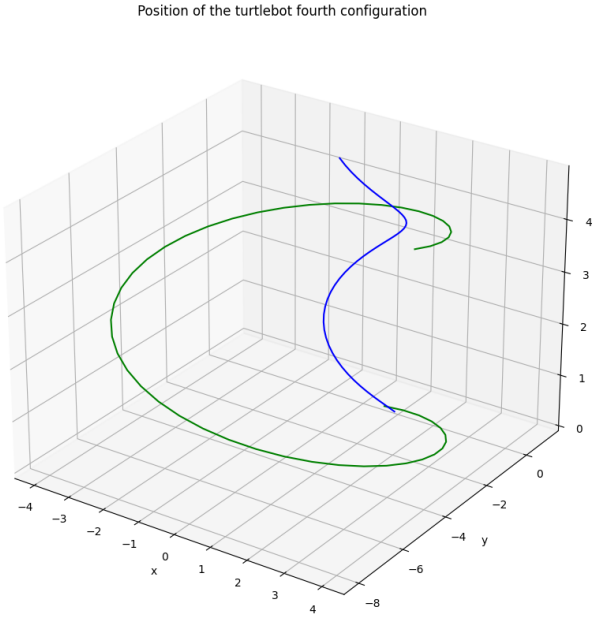

Figure (7) Turtlebot positions with: $w_1 = 4$ and $w_2 = -2$

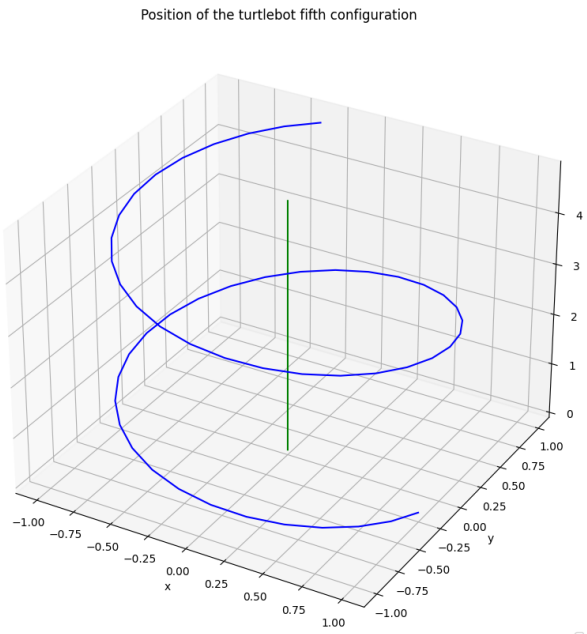

Figure (8) Turtlebot positions with: $w_1 = 4$ and $w_2 = -4$

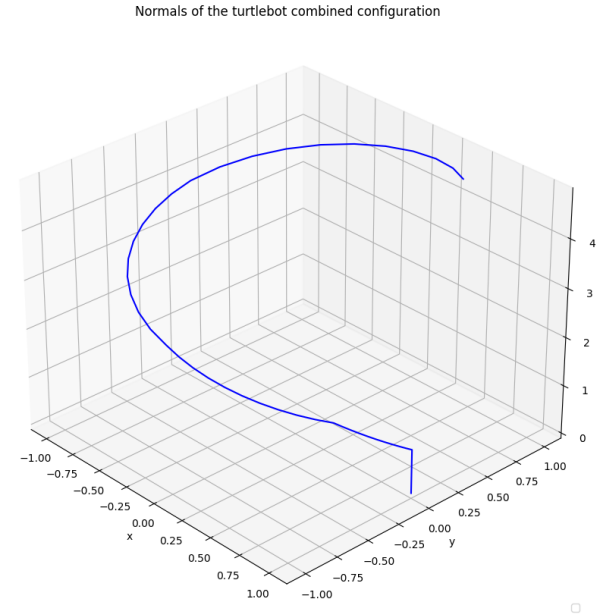

Figure (9) Turtlebot normals with all following configurations

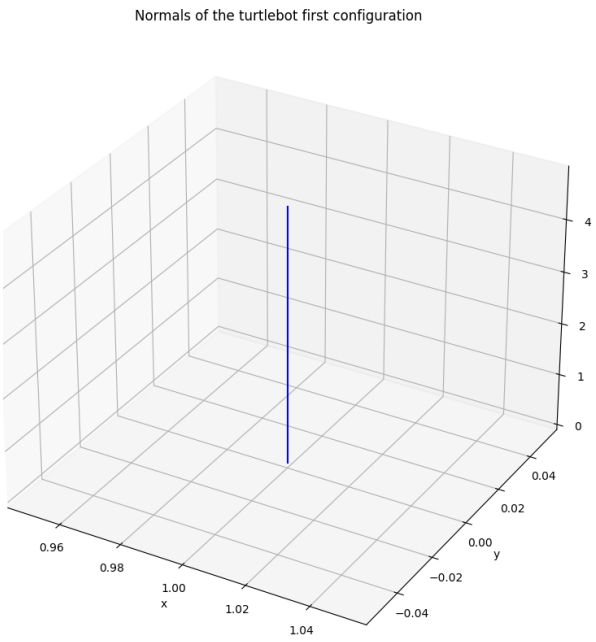

Figure (10) Turtlebot normals with: $w_1 = w_2 = 4$

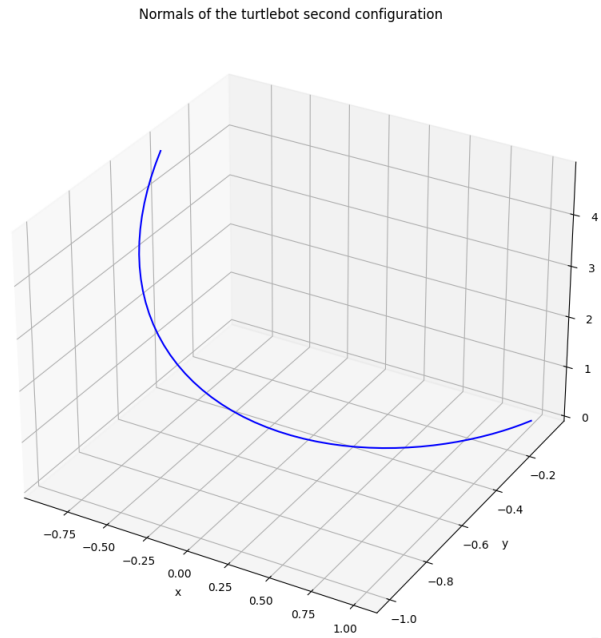

Figure (11) Turtlebot normals with: $w_1 = 4$ and $w_2 = 2$

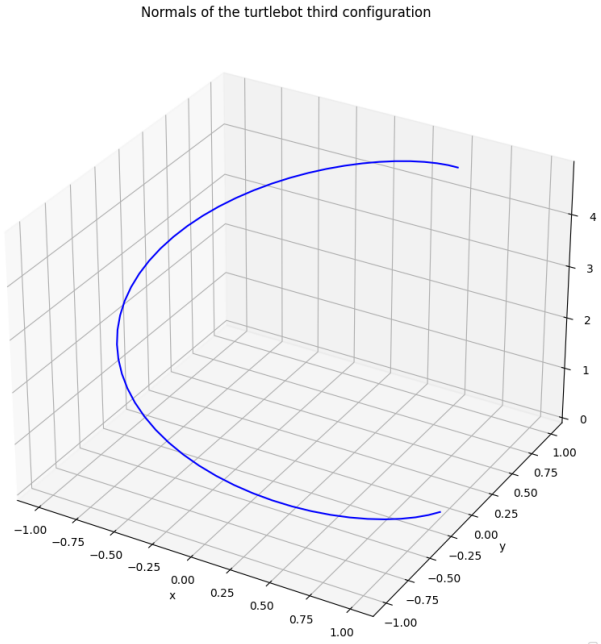

Figure (12) Turtlebot normals with: $w_1 = 4$ and $w_2 = 0$

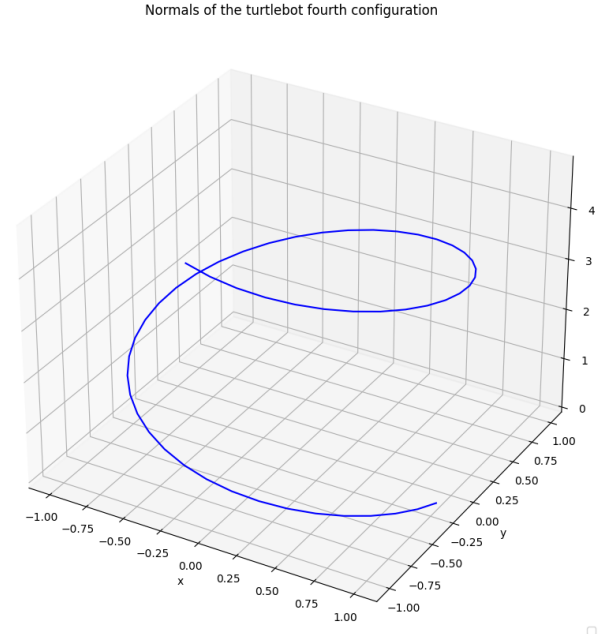

Figure (13) Turtlebot normals with: $w_1 = 4$ and $w_2 = -2$

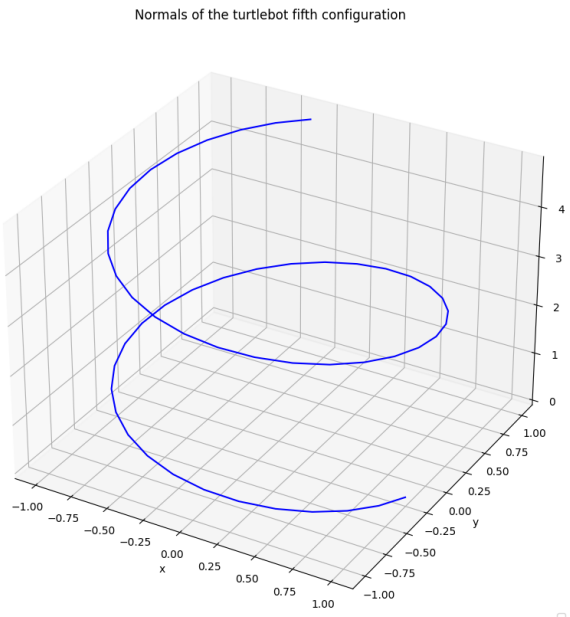

Figure (14) Turtlebot normals with: $w_1 = 4$ and $w_2 = -4$

References

[1] Eck, D.J., 2021. Introduction to Computer Graphics