Obstacle Avoidance

**Descrpition**

This test focuses on evaluating the performance of obstacle avoidance functionality. The expected obstacle avoidance system enables the robot to avoid the obstacle on the path regardless of how the obstacle is placed, the angle the robot is facing.

In this test we will change the position of the obstacle relative to the robot to examine the robot to limits and find out the case that our obstacle avoidance system does not work. For example, we will place an obstacle on the navigation path and place another obstacle near the one on the path. In this condition, the robot is expected to not only avoid the obstacle on the path, but also avoid the obstacle closed to the former obstacle when avoiding the former obstacle.

**Test 1**

**Date :** 2021/3/30

**Tester:** Shichang Zhang

**Author:** Shichang Zhang

**Hardware version:** 1.5 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 1.7 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Determine whether the robot can correctly detect the obstacle and avoid it during travelling to the next waypoint in the navigation process. (Normal Condition)

**Test Procedure:**

We will place the robot at a starting waypoint and we want the robot to move to a destination waypoint. We will place an obstacle somewhere. We expect the robot to successfully detect the obstacle and avoid hitting it. Then the robot is also expected to accurately travel to the next waypoint.

1. The robot is placed at the start waypoint (2,4), pointing to 180°.
2. Set the forward speed of the robot to be 500, rotate speed to be 200.
3. Place the obstacle somewhere. (The detailed position will be shown in Test Data part)
4. Pass the next waypoint as input data.
5. Start the odometer, set the odometer value as the input waypoint.
6. Call the travelTo() method to travel to the next waypoint.
7. Record whether the robot hits the overpass.
8. Record whether the robot successfully stops at the input end waypoint.
9. Stop the program.
10. Print the final translation value indicated by the webot.

**Test Data:**

|  |  |  |
| --- | --- | --- |
| Trial# | destination (ft,ft) | Obstacle Position |
| 1 | (4,2) | Figure 1.1 The navigation path of trial 1 |
| 2 | (5,1) | Figure 1.2 The navigation path of trial 2 |
| 3 | (7,2) | Figure 1.3 The navigation path of trial 3 |
| 4 | (2,1) | Figure 1.4 The navigation path of trial 4 |
| 5 | (9,2) | Figure 1.5 The navigation path of trial 5 |

**Expected Result:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) |
| 1 | No | Yes | (1.2192,0.6096) |
| 2 | No | Yes | (1.5240,0.3048) |
| 3 | No | Yes | (2.1336,0.6096) |
| 4 | No | Yes | (0.6096,0.3048) |
| 5 | No | Yes | (2.7432,0.6096) |

**Test Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) | Error (m,m) |
| 1 | No | Yes | (1.2204,0.6132) | (0.0012,0.0036) |
| 2 | No | Yes | (1.5032,0.2996) | (0.0208,0.0052) |
| 3 | No | No | (2.1177,0.3157) | (-0.0159,-0.2939) |
| 4 | No | Yes | (0.6165,0.3110) | (0.0069,0.0062) |
| 5 | No | Yes | (2.7259,0.5917) | (0.0173,0.0179) |

**Test Report:**

The test is performed 5 times for different destination waypoints and different obstacle placements. The pass rate is 80.0%. We expect the robot to avoid the obstacle and travel to the destination. From the tested output, we failed in trial 3. Actually in this trial, the robot successfully avoided the obstacle. But after avoiding the obstacle, the angle error of the robot was enlarged and it caused the translation value stored in the odometer to be not accurate. When the robot tried to localize to the next waypoint, due to the deviation, the robot failed to localize to the destination (7,2) but localized to the point under the destination (7,1). Another four trials show that the robot can successfully avoid obstacles in normal cases. However, compared to direct travel to the destination waypoint, the errors produced increase considerably. But the errors are still in tolerable range. Overall, the robot is able to detect the obstacle, avoid it, and go to the next waypoint but with larger errors than the condition with no obstacle.

**Conclusion:** Conditionally Pass

**Action:** Improve the accuracy after avoiding the obstacle. Test edge cases.

**Distribution:** Software Development

**Test 2**

**Date :** 2021/4/1

**Tester:** Shichang Zhang

**Author:** Shichang Zhang

**Hardware version:** 1.5 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 1.7 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Determine whether the robot can correctly detect the obstacle and avoid it during travelling to the next waypoint in the navigation process. (Edge Cases)

**Test Procedure:**

We will place the robot at a starting waypoint and we want the robot to move to a destination waypoint. We will place an obstacle somewhere. We expect the robot to successfully detect the obstacle and avoid hitting it. Then the robot is also expected to accurately travel to the next waypoint.

1. The robot is placed at the start waypoint (1,1), pointing to 0°.
2. Set the forward speed of the robot to be 500, rotate speed to be 200.
3. Place the obstacle somewhere. (The detailed position will be shown in Test Data part)
4. Pass the next waypoint as input data.
5. Start the odometer, set the odometer value as the input waypoint.
6. Call the travelTo() method to travel to the next waypoint.
7. Record whether the robot hits the overpass.
8. Record whether the robot successfully stops at the input end waypoint.
9. Stop the program.
10. Print the final translation value indicated by the webot.

**Test Data:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Destination (ft,ft) | Obstacle Position | Comment |
| 1 | (6,1) | Figure 2.1 The navigation path of trial 1 | The obstacle is placed near to the start point. The robot should successfully detect the obstacle and immediately after turning to the correct angle. |
| 2 | (9,1) | Figure 2.2 The navigation path of trial 2 | The obstacle is placed near to the end point. And the obstacle is relatively far from the robot. The robot should know there is an obstacle on the path even though there is a large distance between them. |
| 3 | (6,1) | Figure 2.3 The navigation path of trial 3 | The obstacle is placed near to the direct path, but not exactly on the path. The robot cannot direct travel to the waypoint. The robot should know that there is an obstacle obstructing it from directly travelling even though the obstacle is on the path. . |
| 4 | (6,1) | Figure 2.4 The navigation path of trial 3 | One obstacle is placed on the path and another one is placed near it. The robot should be able to avoid other obstacles while avoiding the obstacle on the path. |
| 5 | (6,1) | Figure 2.5 The navigation path of trial 5 | Similar to trial 4, one obstacle is placed on the travel path and the other one is on the avoid path. The robot is expected to go backward to avoid all these obstacles and travel to the end point. |

**Expected Result:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) |
| 1 | No | Yes | (1.8288,0.3048) |
| 2 | No | Yes | (2.7432,0.3048) |
| 3 | No | Yes | (1.8288,0.3048) |
| 4 | No | Yes | (1.8288,0.3048) |
| 5 | No | Yes | (1.8288,0.3048) |

**Test Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) | Error (m,m) |
| 1 | No | Yes | (1.8276,0.3035) | (0.0012,-0.0013) |
| 2 | Yes | No | (2.2838,0.3067) | (-0.4594,0.0019) |
| 3 | Yes | No | (1.6046,0.4725) | (-0.2242,0.1677) |
| 4 | Yes | No | (1.1366,0.7231) | (-0.6922,0.4183) |
| 5 | Yes | No | (0.8326,0.5877) | (-0.9962,0.2829) |

**Test Report:**

The test is performed 5 times for different destination waypoints and different obstacle placements. The pass rate is 0%. We expect the robot to avoid the obstacle and travel to the destination. From the tested output, we saw that trial 1 succeeded. In trial 1, the robot immediately detected the obstacle and tried to avoid it. And finally it successfully went to the next waypoint. In trial 2, the robot did not detect the obstacle in time. The robot actually successfully detects the robot, but at that time, it is too close to the obstacle so that there is no space for turning. So the robot hit the obstacle and crashed. In trial 3, the robot failed to detect the obstacle until it hit the obstacle. So the robot crashed. In trial 4, the robot succeeded in detecting the obstacle on the path and trying to avoid it. But when avoiding that obstacle, the robot did not detect there was also an obstacle on the avoid path. So the robot hit that obstacle and crashed. In trial 5, similarly, the robot did not detect the obstacle on the avoid path so that it hit the obstacle and crashed. Overall, the robot failed to handle the edge cases. It can hardly detect the obstacles with special placements.

**Conclusion:** Fail

**Action:** Improve the avoid obstacle algorithm to let it be more sensitive.

**Distribution:** Software Development

**Test 3**

**Date :** 2021/4/9

**Tester:** Shichang Zhang

**Author:** Shichang Zhang

**Hardware version:** 1.5 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

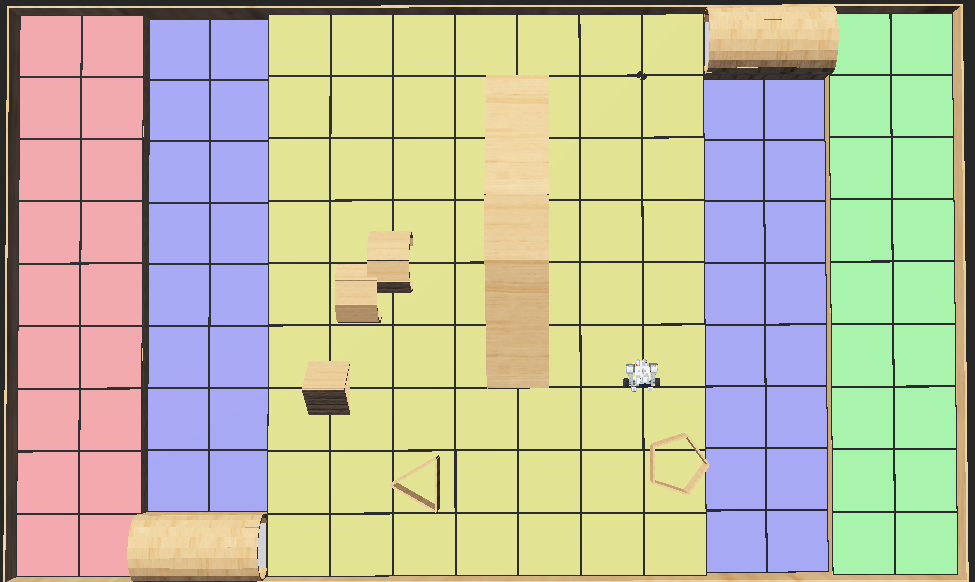
**Software version:** 2.1 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Determine whether the robot can correctly detect the obstacle and avoid it during travelling to the next waypoint in the navigation process. (Single Obstacle Edge Cases)

**Test Procedure:**

Test map:

****

We will place the robot at a starting waypoint and we want the robot to move to a destination waypoint. We will place an obstacle somewhere. We expect the robot to successfully detect the obstacle and avoid hitting it. Then the robot is also expected to accurately travel to the next waypoint.

1. The robot is placed at the start waypoint (10,1), pointing to 90°.
2. Set the forward speed of the robot to be 500, rotate speed to be 200.
3. Place the obstacle somewhere. (The detailed position will be shown in Test Data part)
4. Pass the next waypoint as input data.
5. Start the odometer, set the odometer value as the input waypoint.
6. Call the travelTo() method to travel to the next waypoint.
7. Record whether the robot hits the overpass.
8. Record whether the robot successfully stops at the input end waypoint.
9. Stop the program.
10. Print the final translation value indicated by the webot.

**Test Data:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Destination (ft,ft) | Obstacle Position | Comment |
| 1 | (10,4) | Figure 3.1 The navigation path of trial 1 | The obstacle is placed near to the start point. The obstacle is not exactly on the path, but the robot will hit the obstacle if it directly travels to the waypoint. The robot should successfully detect the obstacle immediately after turning to the correct angle and avoid it. |
| 2 | (10,4) | Figure 3.2 The navigation path of trial 2 | Similar conditions for trial 1, but the obstacle is placed at the left of the path. |
| 3 | (10,7) | Figure 3.3 The navigation path of trial 3 | The obstacle is placed near to the end point. And the obstacle is relatively far from the robot. The robot should know there is an obstacle on the path even though there is a large distance between them. |

**Expected Result:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) |
| 1 | No | Yes | (3.,0480,1.2192) |
| 2 | No | Yes | (3.,0480,1.2192) |
| 3 | No | Yes | (3.,0480,2.1336) |

**Test Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) | Error (m,m) |
| 1 | No | Yes | (3.,0521,1.2241) | (0.0041,0.0049) |
| 2 | No | Yes | (3.,0533,1.2220) | (0.0053,0.0028) |
| 3 | No | Yes | (3.,0517,1.2268) | (0.0037,0.0076) |

**Test Report:**

The test is performed 3 times for different destination waypoints and different single obstacle placements. The pass rate is 100%. We expect the robot to avoid the obstacle and travel to the destination. From the tested output, we observe that the robot is able to handle the conditions listed in the Test Data. For trial 1, it successfully detected that there is an obstacle on the right immediately after rotation. For trial 2, similarly, since the robot now could handle the condition that the obstacle is not on the front ,but beside the path, it found the obstacle on the left and called the corresponding algorithm to avoid it. Trial 3 tested the condition that we failed in the Test2’s trial 2. Last time it was too late for the robot to detect the obstacle, it had no space to turn. This time the robot detected the obstacle in time, so it just normally ran the avoid functions and reached the next waypoint. Overall, the robot now can handle the single obstacle edge cases.

**Conclusion:** Pass

**Action:** Test multiple obstacle cases.

**Distribution:** Software Development

**Test 4**

**Date :** 2021/4/10

**Tester:** Shichang Zhang

**Author:** Shichang Zhang

**Hardware version:** 1.5 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

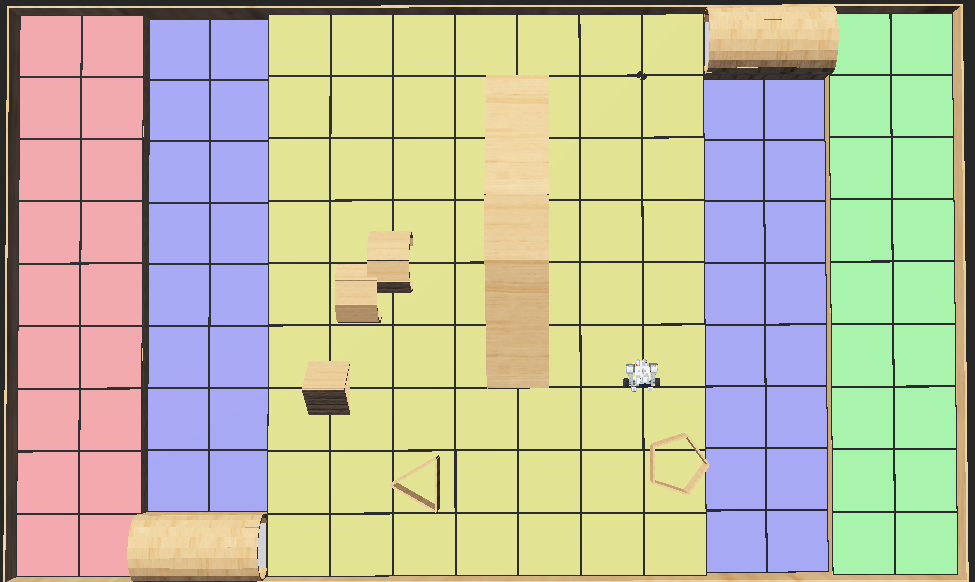
**Software version:** 2.1 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Determine whether the robot can correctly detect the obstacle and avoid it during travelling to the next waypoint in the navigation process. (Single Obstacle Edge Cases)

**Test Procedure:**

Test map:

****

We will place the robot at a starting waypoint and we want the robot to move to a destination waypoint. We will place an obstacle somewhere. We expect the robot to successfully detect the obstacle and avoid hitting it. Then the robot is also expected to accurately travel to the next waypoint.

1. The robot is placed at the start waypoint (10,1), pointing to 90°.
2. Set the forward speed of the robot to be 500, rotate speed to be 200.
3. Place the obstacle somewhere. (The detailed position will be shown in Test Data part)
4. Pass the next waypoint as input data.
5. Start the odometer, set the odometer value as the input waypoint.
6. Call the travelTo() method to travel to the next waypoint.
7. Record whether the robot hits the overpass.
8. Record whether the robot successfully stops at the input end waypoint.
9. Stop the program.
10. Print the final translation value indicated by the webot.

**Test Data:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Destination (ft,ft) | Obstacle Position | Comment |
| 1 | (10,5) | Figure 4.1 The navigation path of trial 1 | Two obstacles are put in front of the robot with almost the same y-coordinate. Then the robot now does not need to avoid, just directly travel to the waypoint. This requires that the robot should not call the avoid obstacle algorithms by mistake. |
| 2 | (10,6) | Figure 4.2 The navigation path of trial 2 | Two obstacles are put in front of the robot with almost the same y-coordinate and one of them is almost placed on the direct travel path. The robot should accurately calculate the path and avoid the obstacles. |
| 3 | (10,5) | Figure 4.3 The navigation path of trial 3 | Two obstacles are placed in front of the robot with almost the same y-coordinate. The space between them cannot allow the robot to pass. The robot should detect the obstacles and figure out the proper avoid path. |
| 4 | (10,5) | Figure 4.4 The navigation path of trial 4 | Two obstacles are placed in front of the robot. The space between them is slightly larger than the robot’s base width. The robot should accurately detect the obstacle position and calculate the path. |
| 5 | (10,5) | Figure 4.5 The navigation path of trial 5 | One obstacle is placed in front of the robot, and the other one is placed on the left of the robot. The right side of the path is the sea. The robot should design a proper avoid path to reach the next waypoint. |
| 6 | (10.,5) | Figure 4.6 The navigation path of trial 6 | One obstacle is placed in front of the robot. Two obstacles are placed closely at the left of the path. The right side of the path is the sea. The robot should continuously avoid the obstacle on the left side when avoiding the obstacle on the path. |
| 7 | (10,5) | Figure 4.7 The navigation path of trial 7 | One obstacle is placed in front of the robot. The other one is placed on the avoid path of the first obstacle. The robot should avoid the path shown in the purple line. |

**Expected Result:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) |
| 1 | No | Yes | (3.,0480,1.5240) |
| 2 | No | Yes | (3.,0480,1.8288) |
| 3 | No | Yes | (3.,0480,1.5240) |
| 4 | No | Yes | (3.,0480,1.5240) |
| 5 | No | Yes | (3.,0480,1.5240) |
| 6 | No | Yes | (3.,0480,1.5240) |
| 7 | No | Yes | (3.,0480,1.5240) |

**Test Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trial# | Hit Obstacle | Stop at Destination | Translation (m,m) | Error (m,m) |
| 1 | Yes | No | (3.4503,1.7143) | (0.4023,1903) |
| 2 | Yes | No | (2.8804,1.3503) | (0.2480,0.4785) |
| 3 | Yes | No | (2.6609,0.5304) | (0.3871,0.9936) |
| 4 | No | Yes | (3.0466,1.5270) | (0.0014,0.0030) |
| 5 | Yes | No | (3.4442,0.7407) | (0.3962,0.7833) |
| 6 | Yes | No | (3.4464,1.0209) | (0.3984,0.5031) |
| 7 | Yes | No | (2.9474,1.4356) | (0.1006,0.0884) |

**Test Report:**

The test is performed 7 times for different destination waypoints and different single obstacle placements. The pass rate is 14.3%. The only passed trial was trial 4. In trial 1, the robot called the avoid algorithms by mistake. So it hit the obstacle and crashed. For trial 2 and 3, the robot failed to find an accurate path to pass through the obstacles. When the robot tried to avoid the obstacle on the path, it hit the other obstacle. For trial 5 and 6, the robot realized that there were obstacles in front of it and on the left of it. And the ultrasonic sensor said there was no obstacle on the right so the robot jumped into the sea. For trial 7, the robot successfully avoided the first obstacle, but the robot hit the first obstacle when avoiding the second obstacle. Overall, the robot currently is not capable of handling complex cases. We need to add more functions to the object avoidance.

**Conclusion:** Failed

**Action:** Try to increase the accuracy of detecting obstacles (very hard, may be updated latest). Add detect sea algorithms (maybe by checking whether the point will be outside of the island). Add the algorithms that can handle the condition that obstacles occur on the avoid path.

**Distribution:** Software Development