Path Manager

**Test 1**

**Date:** 2021/03/24

**Tester:** Lide Cui

**Author:** Lide Cui

**Hardware version:** 1.3 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 1.4 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:** Test the capability of the robot to manage waypoints and identify special points.

**Test Procedure:** We assume that the robot has finished localization and safely across the bridge. The robot will run a PathManager method to rearrange the waypoints so that it could start from the closest point. At the same time, the robot will find two points – the one that the robot will prepare to go over the overpass, the other that the robot will go under the overpass.

**Expected Result:**

The robot should rearrange the waypoints according to its current location and find two special waypoints.

**Test Data/Result:**

Note. Only critical points are shown in the waypoints list.

Table 1 Test waypoints arrangement

|  |  |
| --- | --- |
| Test Inputs waypoints list, curPos | Outputs |
| List ={(1, 4), …}, curPos=(0.5, 5) | Success |
| List ={(1, 4), …, (14, 4), …}, curPos=(14.5, 5) | Success |
| List={(1,4),…, (2, 3),…}, curPos=(1,1) | Success |
| List={(1,4),…, (13, 2),…}, curPos=(14,1) | Success |

|  |  |
| --- | --- |
| Test Inputs OP, GO\_A, GO\_B | Outputs |
| OP=(5,4)-(9,2), GO\_A=(6,5), GO\_B=(9,1) | Success |
| OP=(5,4)-(9,2), GO\_A=(6,5), GO\_B =(8,2) | Fail (one literal is not satisfied) |
| OP=(5,4)-(5,2), GO\_A=(6,5), GO\_B =(8,4) | Fail (vertical overpass cause arctan error) |
| OP=(5,4)-(9,2), GO\_A=(10,5), GO\_B =(12,4) | Fail (wrong pass satisfies conditions) |

|  |  |
| --- | --- |
| Test Inputs OP, GU\_A, GU\_B | Outputs |
| OP=(5,4)-(9,2), GU\_A=(8,4), GU\_B=(4,1) | Success |
| OP=(5,4)-(9,2), GU\_A =(9,4), GU\_B =(8,1) | Fail (two line don’t across) |
| OP=(5,4)-(5,2), GU\_A =(8,4,5), GU\_B =(4,1) | Success |
| OP=(5,4)-(9,2), GU\_A =(6,5), GU\_B =(9,1) | Fail (it confuses go over and go under point) |

**Test Report:**

After doing tests, we find that the robot can correctly rearrange waypoints according to its location. But it was pretty bad at identifying special points, especially when the bridge is vertical or has an odd relative position to the way point.

**Conclusion:** The algorithm to determine the special point needs a lot of refinement.

**Action:** For the go over method, we could add a special case when the bridge is vertical and remove the condition that the distance between GO\_A and GO\_B should be greater than overpass length, instead, we could calculate the distances of GO\_A and OP\_A, GO\_B and OP\_B.

For the go under method, we could keep the intersection literal because the special case when two lines do not cross is quite unusual. Furthermore, we could add a condition to determine the ratio of slopes of GU and OP. If the ratio < 0, then choose it as the GU point, otherwise, drop it.

**Distribution:** Software Development

**Test 2**

**Date:** 2021/4/5

**Tester:** Junjian Chen

**Author:** Junjian Chen

**Hardware version:** 1.3 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 1.8 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:** To find out the capability of the path manager to calculate the correct “overpassStartLocalizationPoint” and “overpassEndLocalizationPoint”.

**Test Procedure:**

1. Set the team color to RED
2. Set two point as endpointA and endpointB of the overpass
3. Check and record whether the output points “overpassStartLocalizationPoint” and “overpassEndLocalizationPoint” match the expected points.

An example of calculation is shown in Figure 1. Endpoint A and B of the overpass is (5,4) and (9,2). If the team color is red, by extending the line (extending lines shown in blue color) between these two points, we can get two intersects with the grids near two endpoints. In the grid near A, we will get overpassStartLocalizationPoint (4,4.5). In the grid near B, we will get overpassEndLocalizationPoint (10,1.5).

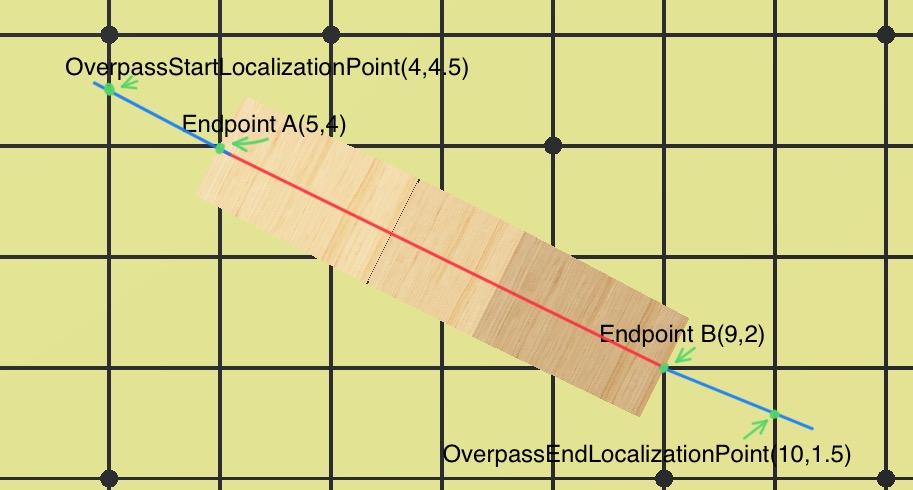


Figure 1: Example of Calculation of expected points

**Test Data:**

|  |  |  |
| --- | --- | --- |
| Trial# | Endpoint A | Endpoint B |
| 1 | (5,4) | (9,2) |
| 2 | (5,5) | (9,2) |
| 3 | (5,3) | (9,2) |
| 4 | (3,4) | (10,2) |
| 5 | (4,2) | (9,4) |
| 6 | (5,2) | (8,4) |
| 7 | (5,2) | (12,3) |
| 8 | (5,2) | (9,2) |
| 9 | (5,2) | (7,6) |
| 10 | (5,2) | (7,5) |
| 11 | (5,2) | (7,4) |
| 12 | (5,2) | (8,7) |

**Expected Result:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trial# | Endpoint A | Endpoint B | Expected overpassStartLocalizationPoint | Expected  overpassEndLocalizationPoint |
| 1 | (5,4) | (9,2) | (4,4.5) | (10,1.5) |
| 2 | (5,5) | (9,2) | (4,5.75) | (10,1.25) |
| 3 | (5,3) | (9,2) | (4,3.25) | (10,1.75) |
| 4 | (3,4) | (10,2) | (2,4.27) | (11,1.71) |
| 5 | (4,2) | (9,4) | (3,1.6) | (10,4.4) |
| 6 | (5,2) | (8,4) | (4,1.33) | (9,4.67) |
| 7 | (5,2) | (12,3) | (4,1.86) | (13,3.14) |
| 8 | (5,2) | (9,2) | (4,2) | (10,2) |
| 9 | (5,2) | (7,6) | (4.5,1) | (7.5,7) |
| 10 | (5,2) | (7,5) | (4.33,1) | (7.67,6) |
| 11 | (5,2) | (7,4) | (4,1) | (8,5) |
| 12 | (5,2) | (8,7) | (4.4,1) | (8.6,8) |

**Test Results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trial# | overpassStartLocalizationPoint | overpassEndLocalizationPoint | Error of overpassStartLocalizationPoint | Error of  overpassEndLocalizationPoint | Pass/Fail |
| 1 | (4,4.5) | (10,1.5) | 0 | 0 | Pass |
| 2 | (4,5.75) | (10,1.25) | 0 | 0 | Pass |
| 3 | (4,3.25) | (10,1.75) | 0 | 0 | Pass |
| 4 | (2,4.29) | (11,1.73) | 0.02 | 0.02 | Pass |
| 5 | (3,1.6) | (10,4.4) | 0 | 0 | Pass |
| 6 | (4,1.33) | (9,4.67) | 0 | 0 | Pass |
| 7 | (4,1.86) | (13,3.14) | 0 | 0 | Pass |
| 8 | (4,2) | (10,2) | 0 | 0 | Pass |
| 9 | (4.5,1) | (7.5,7) | 0 | 0 | Pass |
| 10 | (4.33,1) | (7.67,6) | 0 | 0 | Pass |
| 11 | (4,1) | (8,5) | 0 | 0 | Pass |
| 12 | (4.4,1) | (8.6,8) | 0 | 0 | Pass |

**Test Report:**

Pass Rate:100%

The algorithm can perfectly calculate two localization points. Among 12 trials, 11 trials have 0 error and only one trial has a tiny error of 0.02 which is maybe caused by the default of Math Class of Java and it does not affect the navigation at all.

**Conclusion:** The algorithm to calculate “overpassStartLocalizationPoint” and “overpassEndLocalizationPoint” passes the test.

**Action:** None.

**Distribution:** Software Development

**Test 3**

**Date :** 2021/4/5

**Tester:** Shichang Zhang

**Author:** Shichang Zhang

**Hardware version:** 1.3 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 2.0 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Determine whether the path manager can tell the robot the correct waypoints where the robot is expected to go over or under the overpass. (red team)

**Test Procedure:**

We will pass a list of waypoints and the coordinates of the overpass endpoints to the robot. We expect that the path manager is able to calculate at which point the robot should go over or under the overpass.

1. Pass the input waypoints to the robot.
2. Pass the input overpass endpoints to the robot.
3. Call the path manager’s arrangePoints() method.
4. Print the points at which the robot should perform overpass related actions.
5. Stop the program.

**Test Data:**

|  |  |  |
| --- | --- | --- |
| Trial# | Waypoints (ft,ft) | Overpass endpoints (ft,ft) |
| 1 | {(1,4),(4,5),(6,5),(9,1),(11,1),(14,2),(14,4),(11,5),(8,4),(4,1)} | (5,4),(9,2) |
| Figure 3.1 The planned path on the main island of trial 1 | |
| 2 | {(1,4),(4,5),(5,5),(9,1),(11,1),(14,2),(14,4),(11,5),(9,5),(4,1)} | (6,1),(8,5) |
| Figure 3.2 The planned path on the main island of trial 2 | |
| 3 | {(1,4),(4,5),(6,5),(11,1),(14,2),(13,4),(9,1),(4,1)} | (10,1),(12,5) |
| Figure 3.3 The planned path on the main island of trial 3 | |
| 4 | {(1,4),(4,5),(6,5),(8,4),(14,2),(14,4),(11,5),(11,1),(9,1),(4,1),(3,3)} | (7.5,3),(12,3) |
| Figure 3.4 The planned path on the main island of trial 4 | |
| 5 | {(1,4),(4,5),(6,5),(10,4),(11,5),(14,4),(11,1),(6,1),(7,3),(4,1)} | (8,1.5),(8,5.5) |
| Figure 3.5 The planned path on the main island of trial 5 | |

**Expected Result:**

|  |  |  |
| --- | --- | --- |
| Trial# | Go over waypoints (ft,ft) | Go under waypoints(ft,ft) |
| 1 | (6,5),(9,1) | (8,4),(4,1) |
| 2 | (9,5),(4,1) | (5,5),(9,1) |
| 3 | (13,5),(9,1) | (8,4),(11,1) |
| 4 | (8,4),(14,2) | (11,5),(11,1) |
| 5 | null,null | (6,5),(10,4) |

**Test Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Go over waypoints (ft,ft) | Go under waypoints(ft,ft) | Pass/Fail |
| 1 | (6,5),(9,1) | (8,4),(4,1) | Pass |
| 2 | (9,5),(4,1) | (5,5),(9,1) | Pass |
| 3 | (13,5),(9,1) | (8,4),(11,1) | Pass |
| 4 | (8,4),(14,2) | (11,5),(11,1) | Pass |
| 5 | null,null | (6,5),(10,4) | Pass |

**Test Report:**

The test is performed 5 times for different overpass positions. The pass rate is 100.0%. We expect the robot can calculate at which point it should perform going over or under the overpass. From the tested output, we can see that the robot successfully worked out these points. Furthermore, in trial 5, we placed the overpass at a special position and passed a list of waypoints that did not require the robot to go over the overpass. The robot successfully knew that and pointed out that the robot did not need to go over the overpass at any waypoint. Overall, the path manager currently can accurately calculate the overpass related waypoints.

**Conclusion:** Pass

**Action:** Test for green team condition.

**Distribution:** software development

**Test 4**

**Date:** 2021/4/6

**Tester:** Junjian Chen

**Author:** Junjian Chen

**Hardware version:** 1.3 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 2.0 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Test whether the Path Manager can correctly arrange the waypoints’ sequence.

**Test Procedure:**

1. Modify the input waypoints
2. Check whether the output waypoint list matches the expected sequence.

**Test Data:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial | Team Color | Input waypoints | Map |
| 1 | RED | [(1, 4), (4, 5), (6, 5), (9, 1), (11, 1), (14, 2), (14, 4), (11, 5), (8, 4), (4, 1)] | Shown below |
| 2 | GREEN |
| Figure 4.1 The planned path on the main island of trial 1,2 | | | |
|
| 3 | RED | [(13, 3), (11, 4), (10, 5), (5, 1), (1, 3), (4, 5), (5, 5), (10, 1), (13, 1), (11, 2)] | Shown below |
| 4 | GREEN |  |
| Figure 4.2 The planned path on the main island of trial 3,4 | | | |
|
| 5 | RED | [ (1, 2), (4, 1), (6, 1), (8, 4), (10, 3), (12, 5), (14, 4), (11, 1), (4, 5),(2, 4)] | Shown below |
| 6 | GREEN |
| Figure 4.4 The planned path on the main island of trial 5,6 | | | |
|
| 7 | RED | [ (10, 1), (6, 5),(3, 5), (2, 3), (4, 3), (2, 1), (8, 4), (9, 5), (11, 4), (12, 3)] | Shown below |
| 8 | GREEN |
| Figure 4.4 The planned path on the main island of trial 7,8 | | | |

**Expected Result:**

|  |  |
| --- | --- |
| Trial# | Waypoints |
| 1 | [(1, 4), (4, 5), (6, 5), (9, 1), (11, 1), (14, 2), (14, 4), (11, 5), (8, 4), (4, 1)] |
| 2 | [(14, 4), (11, 5), (8, 4), (4, 1), (1, 4), (4, 5), (6, 5), (9, 1), (11, 1), (14, 2)] |
| 3 | [(1, 3), (4, 5), (5, 5), (10, 1), (13, 1), (11, 2), (13, 3), (11, 4), (10, 5), (5, 1)] |
| 4 | [(13, 3), (11, 4), (10, 5), (5, 1), (1, 3), (4, 5), (5, 5), (10, 1), (13, 1), (11, 2)] |
| 5 | [(2, 4), (1, 2), (4, 1), (6, 1), (8, 4), (10, 3), (12, 5), (14, 4), (11, 1), (4, 5)] |
| 6 | [(14, 4), (11, 1), (4, 5), (2, 4), (1, 2), (4, 1), (6, 1), (8, 4), (10, 3), (12, 5)] |
| 7 | [(3, 5), (2, 3), (4, 3), (2, 1), (8, 4), (9, 5), (11, 4), (12, 3), (10, 1), (6, 5)] |
| 8 | [(12, 3), (10, 1), (6, 5), (3, 5), (2, 3), (4, 3), (2, 1), (8, 4), (9, 5), (11, 4)] |

**Test Results:**

|  |  |  |
| --- | --- | --- |
| Trial# | Waypoints | Pass/Fail |
| 1 | [(1, 4), (4, 5), (6, 5), (9, 1), (11, 1), (14, 2), (14, 4), (11, 5), (8, 4), (4, 1)] | Pass |
| 2 | [(14, 4), (11, 5), (8, 4), (4, 1), (1, 4), (4, 5), (6, 5), (9, 1), (11, 1), (14, 2)] | Pass |
| 3 | [(1, 3), (4, 5), (5, 5), (10, 1), (13, 1), (11, 2), (13, 3), (11, 4), (10, 5), (5, 1)] | Pass |
| 4 | [(13, 3), (11, 4), (10, 5), (5, 1), (1, 3), (4, 5), (5, 5), (10, 1), (13, 1), (11, 2)] | Pass |
| 5 | [(2, 4), (1, 2), (4, 1), (6, 1), (8, 4), (10, 3), (12, 5), (14, 4), (11, 1), (4, 5)] | Pass |
| 6 | [(14, 4), (11, 1), (4, 5), (2, 4), (1, 2), (4, 1), (6, 1), (8, 4), (10, 3), (12, 5)] | Pass |
| 7 | [(3, 5), (2, 3), (4, 3), (2, 1), (8, 4), (9, 5), (11, 4), (12, 3), (10, 1), (6, 5)] | Pass |
| 8 | [(12, 3), (10, 1), (6, 5), (3, 5), (2, 3), (4, 3), (2, 1), (8, 4), (9, 5), (11, 4)] | Pass |

**Test Report:**

The path manager can successfully arrange the sequence of the waypoints according to the team color.

**Conclusion:** Pass

**Action:** None

**Distribution:** Software Development

**Test 5**

**Date :** 2021/4/6

**Tester:** Shichang Zhang

**Author:** Shichang Zhang

**Hardware version:** 1.3 (in Part 2.5 of [Hardware Document](https://docs.google.com/document/d/11jkA_S_xBqyCbcn2NyMuM-OMDEybDfRy/edit#))

**Software version:** 2.0 (in Part 7.0 of [Software Document](https://docs.google.com/document/d/19JaY5629aUu4Y4rjoQJ-jWyeQLqNSAcr/edit))

**Test Purpose:**

Determine whether the path manager can tell the robot the correct waypoints where the robot is expected to go over or under the overpass. (green team)

**Test Procedure:**

We will pass a list of waypoints and the coordinates of the overpass endpoints to the robot. We expect that the path manager is able to calculate at which point the robot should go over or under the overpass.

1. Pass the input waypoints to the robot.
2. Pass the input overpass endpoints to the robot.
3. Call the path manager’s arrangePoints() method.
4. Print the points at which the robot should perform overpass related actions.
5. Stop the program.

**Test Data:**

|  |  |  |
| --- | --- | --- |
| Trial# | Waypoints (ft,ft) | Overpass endpoints (ft,ft) |
| 1 | {(14,4),(11,5),(8,4),(4,1),(1,4),(4,5),(6,5),(9,1),(11,1),(14,2)} | (5,4),(9,2) |
| Figure 5.1 The planned path on the main island of trial 1 | |
| 2 | {(14,4),(11,5),(11,3),(4,2),(1,4),(4,5),(6,5),(9,1),(11,1),(14,2)} | (5,2),(9,4) |
| Figure 5.2 The planned path on the main island of trial 2 | |
| 3 | {(14,4),(11,5),(9,5),(5,1),(1,4),(4,5),(6,4),(9,1),(11,1),(14,2)} | (6,1),(8,5) |
| Figure 5.3 The planned path on the main island of trial 3 | |
| 4 | {(14,4),(11,5),(6,5),(1,4),(4,2),(11,4),(8,4),(9,1),(11,1),(14,2)} | (5,3),(10,3) |
| Figure 5.4 The planned path on the main island of trial 4 | |
| 5 | {(14,4),(11,5),(9,5),(7,1),(1,4),(4,5),(7,4),(11,3),(11,1),(14,2)} | (8,2),(8,5) |
| Figure 5.5 The planned path on the main island of trial 5 | |

**Expected Result:**

|  |  |  |
| --- | --- | --- |
| Trial# | Go over waypoints (ft,ft) | Go under waypoints(ft,ft) |
| 1 | (6,5),(9,1) | (8,4),(4,1) |
| 2 | (11,3),(4,2) | (6,5),(9,1) |
| 3 | (9,5),(5,1) | (6,4),(9,1) |
| 4 | (4,2),(11,4) | (8,4),(9,1) |
| 5 | (9,5),(7,1) | (7,4),(11,3) |

**Test Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| Trial# | Go over waypoints (ft,ft) | Go under waypoints(ft,ft) | Pass/Fail |
| 1 | (6,5),(9,1) | (8,4),(4,1) | Pass |
| 2 | (11,3),(4,2) | (6,5),(9,1) | Pass |
| 3 | (9,5),(5,1) | (6,4),(9,1) | Pass |
| 4 | (4,2),(11,4) | (8,4),(9,1) | Pass |
| 5 | (9,5),(7,1) | (7,4),(11,3) | Pass |

**Test Report:**

The test is performed 5 times for different overpass positions. The pass rate is 100.0%. We expect the robot can calculate at which point it should perform going over or under the overpass. From the tested output, we can see that the robot successfully worked out these points. In trial 2 and 4, the waypoints that require the robot to go over the overpass is far away from each other. The path manager needs to point out that the robot should perform overpass action there even though the distance between these two waypoints are large. In trial 4 and 5, we tested the condition that the overpass is horizontal and vertical. The algorithm also successfully handles the edge cases. Overall, the path manager currently can accurately calculate the overpass related waypoints.

**Conclusion:** Pass

**Action:** None

**Distribution:** software development