e-Yantra Robotics Competition - 2018

NS Task 1 Report <633>

Author Name(s)	**Tarun Shekher,Anuj Jain,Kapil Gera,Akshat Mishra**
Team ID	**633**
Date	**28/11/2018**

Q1. Describe the path planning algorithm you have chosen.

We have used Dijkstra's algorithm for which a given source node in the graph, finds the shortest path between that node and every other provided that the nodes are reachable from the starting node. The algorithm used here find the shortest possible path from a particular node to a destination node by stopping the algorithm once the the shortest path between the node and the destination node has been determined.This algorithm will continue to run until all of the reachable vertices in a graph have been visited for finding the shortest distance among the nodes.

Q2. Describe the algorithm's specific implementation i.e. how have you implemented it in your task?

First we analysed our given track without considering obstacles in our path.Then we made a adjancy matrix which is weighted and directed i.e having directions.Then we considered one node and assigned different weights to all its neighbouring node. We repeated this process for all the other nodes.Then we made a matrix of 23*23 which was completely based on the weights assigned to different neighbours.Then we gave this matrix input to Digkstras.

Dijkstra's algorithm is used to determine the shortest path from one node in a graph to *every other node* within the same graph data structure, provided that the nodes are reachable from the starting node.
This algorithm will continue to run until all of the reachable vertices in a graph have been visited, which means that we could run Dijkstra's algorithm, find the shortest path between any two reachable nodes, and then we saved the result in a matrix.

Dijkstra's algorithm:

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
Set the initial node as current. Mark all other nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
For the current node, consider all of its neighbors and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

Rules for running Dijkstra's algorithm:

1.Every time that we set out to visit a new node, we will choose the node with the smallest known distance/cost to visit first.

2.Once we've moved to the node we're going to visit, we will check each of its neighboring nodes.

3.For each neighboring node, we'll calculate the distance/cost for the neighboring nodes by summing the cost of the edges that lead to the node we're checking from the starting vertex.

4.Finally, if the distance/cost to a node is less than a known distance, we'll update the shortest distance that we have on file for that vertex.

So finally we apply Dijikstra's on our starting node which gives us the shortest path to our pick up zone.After the bot reaches pickup zone again Dijkstra's is applied to reach the drop zone.After reaching drop zone again Dijikstra's is applied and the process is repeated till all the nuts are dropped. For obstacles, the node just before it, is taken and maximum weight (say 9999) is given to that edge so that there is no chance that in Dijikstra's, that path comes as the shortest path. So the bot follows the best path and avoids the obstacle path.After dropping all the nuts Dijikstra's is again applied on the last dropping node to reach the starting zone.

So our bot completes the given task in the best possible manner.