International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 4, April - 2014

Case-Based Reasoning (CBR) based Path-Planning using Robotic Operating System (ROS)

P. M. Manoj,
M.Tech Embedded Systems,
Karunya University, India.

Mr. C. Mahesh,
Asst. Professor,
Karunya University, India.

***Abstract*--Path planning in robotics has always been a research topic. Several path planning methods are proposed with improved efficiency interms of shortest path finding capability, cost, space and time complexities. State space search and heuristic search methods are the widely adapted methods for path planning. Breadth-First Search, Depth-First Search, Dijikstra's algorithm and A-star algorithm are the techniques used for path planning. Out of which A-star algorithm has proven characteristics of finding the shortest path.**

Case-Based Reasoning (CBR) system implements a simple form of dynamic memory that learns by adding new cases, as and when new problems are encountered and solved. In this paper, Case-Based Reasoning is used for creating a case-base for path-planning. So different paths are stored as cases and when robots are placed in an environment it tries to retrieve the solution from the case-base, but if the environment is new it retains the solution. Thus it helps the Heuristic algorithm to function effectively and reduces the load to the processor which can perform some other task in the mean time.

Keywords-- Path planning; Heuristic search; A-Star algorithm; Case based Reasoning.

I. INTRODUCTION

Path planning is an important issue in robotics as it provides the robot to reach the goal node from a starting node. It is measured by its computational complexity. The feasibility of real-time motion planning is dependent on the accuracy of the map, on robot localization and on the number of obstacles. Topologically, the problem of path planning is related to shortest path problem of finding a route between two nodes in a graph. In algorithms such as A*, informed search method is used. Informed search algorithms are based on a function called Heuristic function, which helps in deciding the path. Heuristic functions exploit domain knowledge to orient the search process towards the desired goal. The objective of using a heuristic function is to improve the efficiency of the solution finding process.

A* algorithm combines the best features of Branch and Bound , Dijikstra's algorithm and Best first search algorithm. Both Branch and Bound algorithm and Dijikstra's algorithm extend the least cost partial solution. Branch and Bound generates a search tree that may have duplicate copies of the same nodes with different costs, while the latter searches over a given graph, keeping exactly one copy of each node and back pointers for the best routes. Though both of them do not have a sense of direction, Best First Search algorithm comes in point. Best First Search algorithm uses a heuristic function to decide which of the candidate node is likely to be closest to the goal, and expands that. However, it does not keep track of the cost incurred to reach the node. Best First Search only looks ahead from the current node, seeking a quick path to the goal, while Branch and Bound only looks behind, keeping track of the best paths found so far. Algorithm A* does both. The advantages of the algorithm A* is that it finds a path even if the graph is infinite and if there exists a path. Also, the found path is the shortest path between the nodes.

Once the A* algorithm is implemented, it is made to provide solutions for different environments. All these solutions are stored as case-bases so that they can be used again if the robot faces a similar problem again, thereby saving the time to calculate the solution again for the same problem which it incurred already. This is achieved using a concept called Case-Based Reasoning that is loaded with many solutions and also, it keeps on storing the solutions whenever new problems are solved. In the mean time, the robot can perform any other useful computation, that is the key point behind this paper.

Case-based method and grid map method are used together in path planning by Maarja Kruusmaa[1], in which case-based system stores the solution and evaluates the traversability. Grid-map method is used to find the solution, whereas case based system is used only when the robot travels in the same path again. In Ashok Goel's paper[2], model-based and case based reasoning are used for path planning. Model based reasoning decomposes the main goal into smaller tasksthe main goal into smaller tasks, such as finding a path from initial location to the neighbourhood towards the goal location. Case-based reasoning derives a path by evaluating the case generated by model-based reasoning. When new problems are found, model based reasoning is alone used and at the same time these solutions are stored in the case base and when the robot experiences the same task again, it makes use of the case-based reasoning to reduce the cost of time required.

II. METHODOLOGY

A. Standard Template Library

Standard Template Libraries(STL) provides a suite of reusable programs, or lines of code, that could be used to increase the programming productivity and quality. STL provides a ready-made set of common classes for C++, such as containers and associative arrays, that can be used with any built-in type and with any user-defined type. It achieves them using templates and also it provides compile-time polymorphism which is more efficient than traditional run-time polymorphism. It has sequence containers and associative containers in which sequence container has vector and associative container has map, which are used in our path planning algorithm. Advantage of using vector container is that it has the ability to resize itself automatically when inserting or erasing an object. Map container allows mapping from one data item(a key) to another(a value) and it is typically implemented using a self-balancing binary search tree. So, an efficient path planning A* algorithm is written using Standard Template Library(STL) in C++.

B. Robotic Operating System:

The Robotic Operating System(ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes and package management. It is based on a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages.

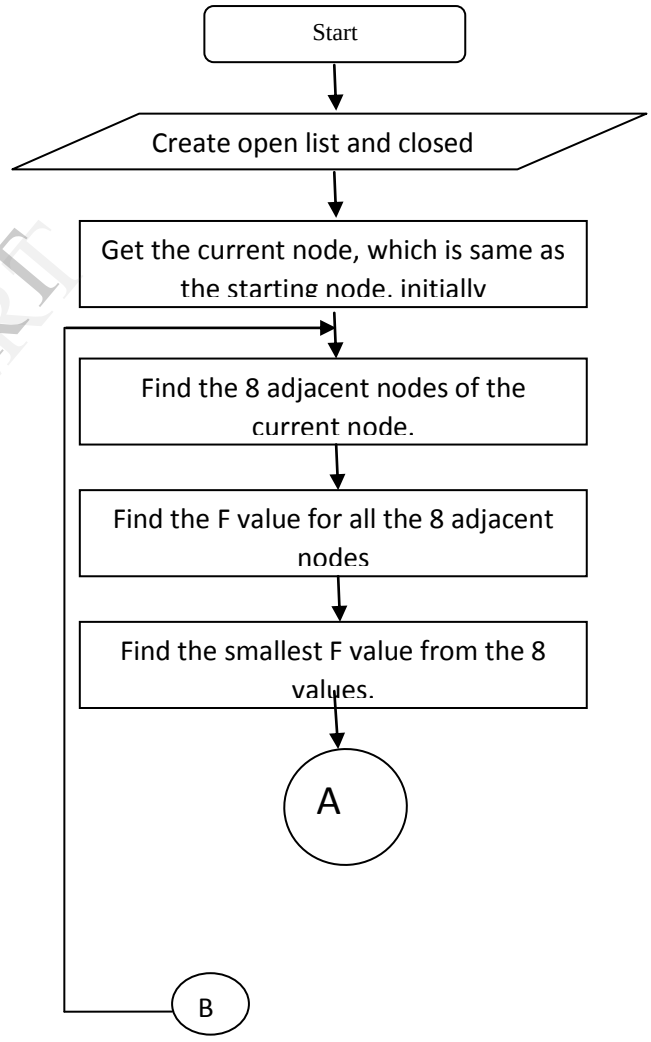
A ROS workspace is created with A* algorithm as a package in it, which is made to act as global planner for the local planners that are already there in the ROS packages. ROS has both 2D and 3D simulators, Stage and Gazebo respectively which is used to visualize the efficiency of the algorithm through a simulated robot.

C. Case-Based Reasoning:

Case-based Reasoning(CBR) system implements a simple form of dynamic memory that learns by adding new cases, as and when new problems are encountered and solved and these cases are related to path planning in this application. So different paths are stored as cases and when robots are placed in an environment it tries to retrieve the solution from the case-base, if the environment is new, it retains the solution from the robot. This feature of this method favors the path-planning algorithm in storing and retrieving the solutions as and when they are necessary, thereby making the processor free from computational complexities which can be used for some other useful processes.

III. IMPLEMENTATION

To verify the functionality of the algorithm, Netbeans IDE software is used in Ubuntu. The steps involved in A* algorithm are, getting the start and the goal nodes from the user, Getting the current node which is same as starting node in the beginning, since 2D grid is used as environment a single node has eight successor nodes which are found, two lists are created, one for openlist and the other for closedlist, all the successor nodes are stored in the openlist, and from which the node with lowest A* estimate function (F value) is taken out and pushed into closedlist. F value is the sum of the cost taken to reach the current node from the starting node and the total cost from current node to the goal node. All these steps are repeated until the goal node is reached. Once the goal is reached, the co-ordinates (way-points) to reach the goal node are generated. The steps mentioned above are summarized in the flowchart in the Fig. 1



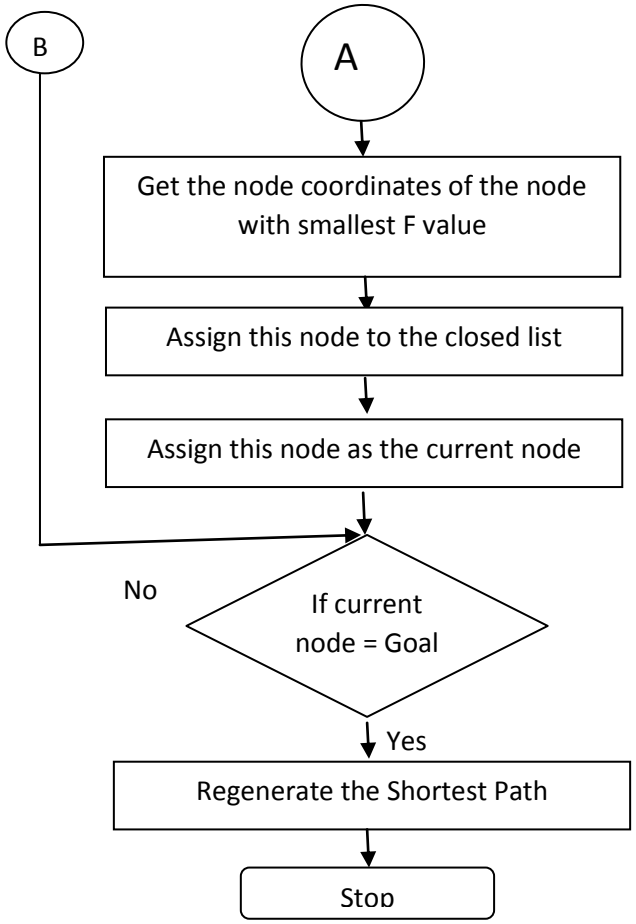


Fig. 1. Flowchart of the algorithm used in coding

Now that the waypoints to the goal are generated, these co-ordinates has to be sent to the global planner of the robot. To achieve this, a Robotic Operating System(ROS) workspace is created with A* as a package in it. ROS provides an environment to create plugins for the workspaces which are already there. So this global planner can be merged with a simulation, having a local planner already. RVIZ launcher is better to visualize the robot in its environment and Stage simulator is used in 2D simulation, as shown in the Fig. 2.

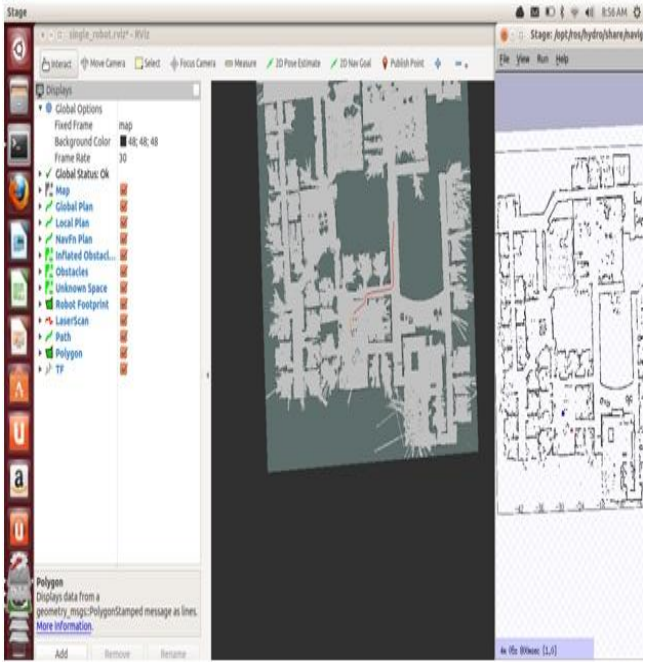


Fig.2 . Simulation output.

4. CONCLUSION:

Thus a highly efficient A* path planning algorithm is implemented using the Standard Template Libraries with map and vector containers as the key role concepts and the same is simulated using Stage simulator and is visualized in RVIZ launcher. Since the coding is done using Standard Template Library, it is completely reusable in any application in which the algorithm fits in and also memory related issues are taken care of by the same since dynamic memory allocation is used in the coding. As a future work, a huge case-base will be created, so that the robot's global planner will be able to give the solution directly from the case-base, instead of calculating the co-ordinates or waypoints.

5. References

1. Subhrajit Bhattacharya, Vijay Kumar and Maxim Likhachev, "Search-based Path Planning with Homotopy Class Constarints", in Proceedings of the Twenty-Sixth AAAI conference on Artificial Intelligence, pp 2097-2099., 2012.
2. Jaroslav Hodal and Jiri Dvorak, "Using Case-Based Reasoning for Mobile Robot Path Planning" in Engineering Mechanics, vol.15, No.3, pp. 181-191., 2008.
3. Maarja Kruusmaa and bertil Svenson, "Combined Map-based and Case-Based Path Planning for Mobile Robot Navigation", in Proceedings of International symposium of Intelligent Robotic Systems, pp 1-6., 1998.
4. Ashok Goel, Michael Donellan, Nancy Vazquez and Todd Callantine, "An Integrated Experience-Based Approach to Navigational Path Planning for

Autonomous Mobile Robots" in IEEE., pp 818-825., 1993.

5. Ashok K. Goel, Todd J.Callantine, Michael W. Donnellam and AAndrez Gomez de Silva Garza, "Case-Based Path Planning: Router", in AAAI Technical Report WS-93-04, pp 162., 1993.