UGC Approved Journal

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 8, August 2017

Robot Path Planning with L * Algorithm for Linear Computational Complexity, Considering Distance and Slope

Helga V Lobo[1], Dr. S.A. Angadi[2]

P.G Student, Department of Computer Science Engineering, Centre for P.G Studies, VTU, Belagavi, India [1]

Professor, Department of Computer Science Engineering, Centre for P.G Studies, VTU, Belagavi, India [2]

Abstract: This paper describes implementation of L* algorithm which provides linear computational complexity in robot path planning which could not be achieved by A*, D* and other predecessors. The proposed work executes the L* algorithm by considering the path slope in every move. The proposed method is implemented using MATLAB software tool. The results of implementation are satisfactory. The average number of times the list is accessed for computing the path in a grid of 9 cells is 6, average time in a grid of 25 cells is 16, in a grid of 100 cells it is 75, and the results are comparable to existing approaches.

Keywords: Robot path planning, linear computational complexity, slope.

I. INTRODUCTION

Path planning is crucial in robotics for the autonomy of the robot. Numerous path planning approaches have been evolved from mid 60's. Dijkstra's, A* and modified versions of A* are some of the graph search algorithms used to find path. Dijkstra's was the earliest algorithm used in path planning [6]. It was aimed to find the shortest path between a given pair of source and destination. The algorithm was zero heuristic algorithm, hence took longer search times as there was no guidance to search. The real time applications could not be addressed by the algorithm, due to longer search times of the algorithms.

A* algorithm [7] was successor of Dijkstra's algorithm which worked well when there was time constraint, as its search was guided by heuristics. A* algorithm uses the knowledge of graph, thereby improving computational efficiency. The algorithm iteratively searches the graph to find a connecting path between given source and destination. A 'dg' cost is associated with each edges of the graph, which represents the distance between the nodes which are connected by that edge.

Thus the path cost between any two nodes is the sum of all 'dg' costs along the path. Search begins with initial node moving towards the goal due to the guidance of heuristic cost and it ends in the goal node. The heuristic cost or 'h' cost considered most commonly is the Euclidean distance from current node to the goal. A 'g' cost is associated with the node, which is the sum of the 'g' cost of the parent of the node and the 'dg' cost of arc joining the parent and the node considered. Algorithm also assigns 'f' cost for each node, which is the sum of the 'g' and 'h' costs of the node. The 'f' costs are stored in an open list stored in memory which represents the nodes in consideration. Need of sorting the open list each time to find a most promising node is the cause for linear-logarithmic complexity [7] of A* algorithm.

These algorithms suffer exponential time complexity with increased size of workspace. This is due to a common feature shared by all these algorithms which requires sorting a list of final costs each time to consider a successor node. This paper proposes a methodology to consider the slopes in a path in every step along the path from source to destination needed to be traversed by the robot. It is required to find the shortest possible path avoiding sharp or steep slopes which cannot be traversed practically. In robot path planning optimal path seeks optimization of several parameters like distance, slope along the path in every step, roughness on land etc.

The problem focused by this paper is improving the time complexity of robot path planner, to bring it to a value lower than that can be provided by existing graph search approaches such as Dijkstra's, A*, and its modified versions of A*, when the workspace is very large. It also considers the path slope in every move. The standard for comparison is taken to be A*, as it is the best representative of existing approaches. Expected result is a path with time complexity lower than existing approaches, with a choice of path with smoother slopes. The average number of times the list is accessed

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 8, August 2017

in a grid of 9 cells is 6, average time in a grid 25 cells is 16, in a grid of 100 cells is 75, and in a grid of 1600 cells it is 739.

The rest of the paper is laid out as follows. Section II presents the literature survey. Section III presents the proposed methodology. Section IV brings up the conclusion.

II. LITERATURE SURVEY

Literature survey reveals that the major path planning approaches used in robot path planning till date for modelling the workspace are roadmap methods, cell decomposition, and potential fields[2][4][5]. In roadmap method the workspace is modelled using a network, whereas the cell decomposition [3] approach models the workspace as grid and the potential field method models the workspace using mathematical function on the entire search space and it is based on the imagination that goal has attractive force on robot whereas the obstacles repel the robot. Modelling of the workspace follows the graph search using methods available like Dijikstra's, A* and modifications of A*. Dijkstra's algorithm is a widely used graph search method for finding shortest path. It searches the shortest path between two given nodes in a graph, and it also extracts the least cost path from all nodes to the source nodes [6]. It is an uninformed search method with no guidance at each step to lead to goal, resulting in longer search time. A* algorithm [7] was conceived in 1968, which uses a heuristic to narrow the search. It is a static algorithm. Whenever there is a change in configuration space, the path found by A* gets invalidated and requires the algorithm to run again [11]. Dynamic A* or D* search was proposed by Stentz [8]. The D* algorithm avoids backtracking and very high cost associated with it. With D* there was least changes to the path due to change in local environment. The focused D* algorithm [10] that puts to gather the ideas of original A* and D*, starts its search from goal moving in direction towards the source. The algorithm can be used for dynamic environments, due its faster path replanning abilities. D* Lite was introduced by Likhatchev and Koenig [9], implements the same behaviour as that of D*. The basis of D* Lite is Lifelong planning A* and not D*.

All these algorithms store final costs of nodes in memory which is given by,

f- cost(node)= g-cost(root, node) + h-cost(node, goal)

These algorithms sort the list containing these final costs, to find most promising node. This sorting procedure causes the time complexity to grow exponentially with increased workspace size.

III.PROPOSED METHODOLOGY

A. Modelling of Workspace

In this particular implementation of robot path planner, the workspace is modelled using cell decomposition. In this modelling of workspace the entire workspace is broken down into finite sized cells. The graph is generated by points picked from every cell, by joining them to points in every neighbouring cell.

Assigning weights to cells

In the grid that is generated by workspace modelling using cell decomposition a weight value is assigned to each grid cell depending on its height from reference cell. Efforts involved in traversing a cell are calculated by multiplying a constant factor to difference in gradient.

Following are the weights assigned to cells in grid,

Wi,j = [Wi,j of origin], if cell referred is in same level of origin cell.
Wi,j > [Wi,j of origin], if cell referred is higher than the origin cell.
Wi,j < [Wi,j of origin], if cell referred is lower than the origin cell.

B. Experimentation

L* Algorithm

The L* algorithm proposed by Adam Niewola et. al [1] maintains sub-lists rather than a single list of final costs. Sub-list 1 will contain the nodes whose final cost is between h0 and h0+df. Sub-list 2 will contain the nodes whose final cost is between h0+df and h0+2df. The first node extracted by the algorithm is the one from first sub-list which is non empty. The nodes are extracted one by one till the goal is found or all the sub-lists get empty. The table1 shows indicative sub-lists. The N^7_4 represents node 4 with its predecessor node 7. The value h0 is the heuristic cost from root node to goal. Heuristic estimate of a node to the goal is given by Euclidian distances. The value of 'df' is taken such that the successor of any node must not appear in its preceding sub-list.

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 8, August 2017

TABLE I THE ARRANGEMENT OF FINAL COSTS INTO SUB LISTS

Range of f-cost			
Sub-list1 h0to h0+df	Sub-list2 h0+df to h0+2df	Sub-list3 h0+2df to h0+3df	.....
N^3_0	N^8_2	N^7_4	.....
.	N^5_1	.	.....
.	.	.	.....
.	.	.	.....

Path generation considering distance and slope

When the path planner need to update the final costs of all its neighbouring nodes, if the adjacent node is at higher slope from the current node the path planner is made to increase the final cost of the neighbour. Hence if there is any steep or sharp slope from the current node to adjacent neighbour, the path planner is made to avoid that path by assigning a very large final cost to that neighbour. The path planner considers a surface which is a plot of distance from one cell to the next vs. weights assigned to each cell. The path planner updates the final cost of each neighbour of current cell considering the slopes in implicit surface while moving from one cell to the other as given below, where 'ds' is change in distance from one cell to next and 'dh' is difference in height from on cell to the next.

Case 1: If dh/ds is +ve and less than or equal to 0.85, the path is acceptable as difference in height with difference in distance is within the proportional limit.
Case 2: If dh/ds is +ve and more than 0.85, the path is rejected as the difference in height with difference in distance is a sharp slope.
Case 3: If dh/ds is -ve and |dh/ds| more than 0.85, the path is rejected as the difference in height with difference in distance is sharp slope.
Case 4: If dh/ds is -ve and |dh/ds| is less than or equal to .85, the path is accepted as difference in height with difference in distance is within proportional limit.

C. Results
Time Comparison
The A* algorithm is taken as standard of comparison as it can be taken as representative for Dijkstra's, A* and other modification due to the common features shared by them. The algorithms are executed for various combinations of source, destinations, obstacle, and terrain. The time taken in terms of list access by A* and L* is presented in the following table.

TABLE III TIME TAKEN IN TERMS OF LIST ACCESS FOR DIFFERENT WORKSPACE SIZE

No of nodes	A*	L*	L* with slopes
25	27	16	25
49	67	49	48
100	145	74	95
169	279	155	172
289	499	271	155
400	687	348	301
529	1054	528	447
625	1098	525	512
784	1368	650	649
1600	2770	739	1090
2500	4461	1932	1932

Following is the graph showing the improvement of L* over A*.

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 8, August 2017

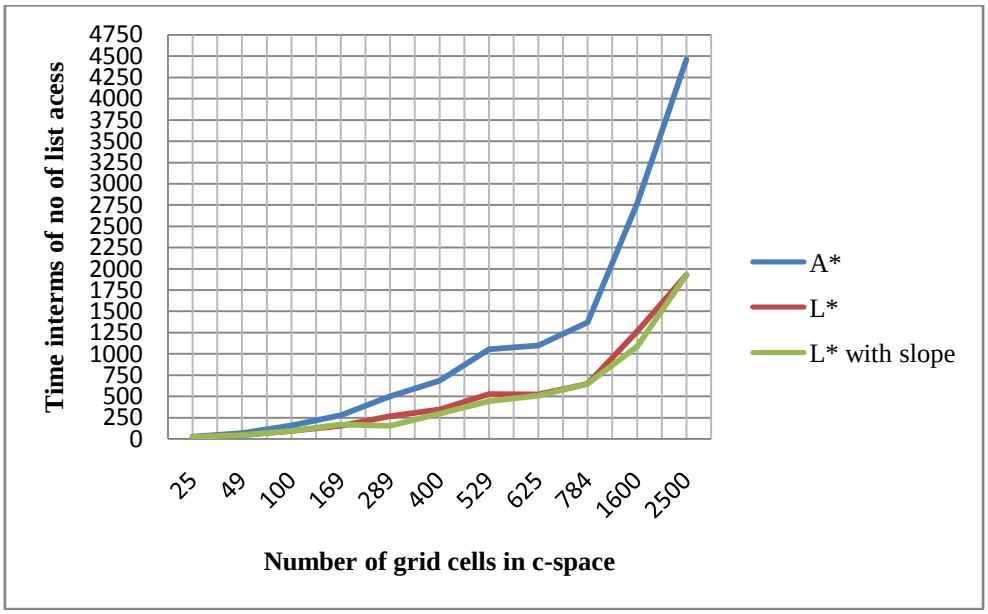

Figure 1: Graph comparing the time taken by A* and L* for different workspace size

Path generated considering the distance and slope

A sample workspace is presented in figure 2 and figure 3. Each cell has a cell number and a value which represents the efforts involved in moving to that cell. This value of effort is obtained by multiplying a constant factor to difference in gradient. The cost to move from one cell to the adjacent vertically and horizontally is considered to be 10 and diagonal cost to be 14. In figure 2 for the given source and destination most obvious shortest path is along the diagonal. And also the slopes (dh/ds) are within the proportional limits as shown by the calculations below.

slope (cell1, cell6) $_{(dh/ds)}$ =|(10-1)/14|=0.64
slope (cell6, cell11) $_{(dh/ds)}$ =|(20-10)/14|=.71
slope (cell11, cell16) $_{(dh/ds)}$ =|(30-20)/14|=0.71
Hence the path along the diagonal is presented as the shortest path by the path planner.

1Source **1**	2 80	3 50	4 60
5 10	6 **38**	7 30	8 60
9 20	10 100	11 **30**	12 50
13 30	14 40	15 50	16 **50** Destination

Figure 2: Path chosen along diagonal as there are no sharp slopes

In figure 3 the slopes along the diagonal are sharper and the path planner bypasses the diagonal choosing smoother slopes as shown by the calculations below.

slope (cell1, cell6) $_{(dh/ds)}$ =|(38-1)/14|=2.6 a sharp slope along diagonal rejects the path
slope(cell6, cell11) $_{(dh/ds)}$ =|(30-38)/14|=0.57
slope(cell11, cell16) $_{(dh/ds)}$ =|(50-30)/14|=1.42 a sharp slope along diagonal rejects the path
slope(cell1, cell5) $_{(dh/ds)}$ =|(8-1)/10|=0.7
slope(cell5, cell9) $_{(dh/ds)}$ =(10-8)/10|=0.2
slope(cell9, cell13) $_{(dh/ds)}$ =|(15-10)/10|=0.5
slope(cell13, cel14) $_{(dh/ds)}$ =|(18-15)/10|=0.3
slope(cell14, cell15) $_{(dh/ds)}$ =|(24-18)/10|=0.6
slope(cell5, cell16) $_{(dh/ds)}$ =|(32-24)/10|=0.8

UGC Approved Journal

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 8, August 2017

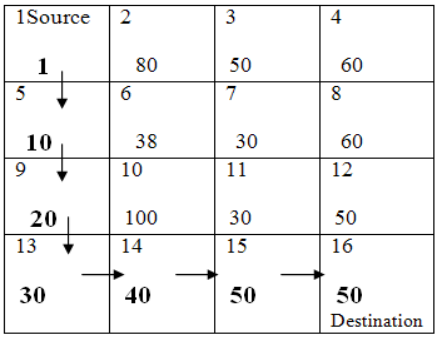

Figure 3: Path planner bypasses the shortest possible path along diagonal to avoid sharp slopes.

IV. CONCLUSIONS

The L* algorithm proves to be more advantageous than A* and its modifications when the workspace size grows large. It keeps the time complexity linear with increasing workspace size. The method presented here to generate path considering the distance and slopes in robot path planning is a very simple approach. This can be used even on the graphs generated by other methods of workspace modelling in robot path planning like road map approaches. The method that used to consider the slope in every step along the path can be extended to consider terrain roughness in a very similar way, where the weights will be assigned based on severity of roughness.

REFERENCES

[1] Adam Niewola, Leszek Podsędkowski, "Nonholonomic Mobile Robot Path Planning with Linear Computational Complexity Graph Searching Algorithm*" , Proceedings of the 10th International Workshop on Robot Motion and Control, Poznan University of Technology, Poznan, Poland, July 6-8, 2015

[2] Sariff, N. and Buniyamin, N.; "An Overview of Autonomous Mobile Robot Path Planning Algorithms", IEEE 4th Student Conference on Research and Development, 2006.

[3] Christian Scheurer & Uwe E. Zimmermann, "Path Planning Method for Palletizing Tasks using Workspace Cell Decomposition," ICRA Communications, 2011.

[4] D. Glavaski, M. Volf and M. Bonkovic, "Mobile Robot Path Planning using Exact Cell Decomposition and Potential Field Methods," WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, vol. 8, no. 9, pp. 789-800, September 2009.

[5] J. N. Tsitsiklis. "Efficient algorithms for globally optimal trajectories," IEEE Transactions on Automatic Control, Volume 40,Issue 9, 1995, pp. 1528-1538.

[6] A. Sedeño-Noda, A. Raith, "A Dijkstra-like method computing all extreme supported non-dominated solutions of the biobjective shortest path problem," Computers & Operations Research. Volume 57, May 2015 (available online December 2014), pp. 83–94

[7] J.-C. Latombe, "Robot Motion Planning," Kluwer Academic Publishers, 1991.

[8] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments," Proceedings of the IEEE International Conference on Robotics and Automation, vol.4, 8-13 May 1994, pp. 3310–3317.

[9] J. Guo, L. Liu, Q. Liu, Y. Qu, "An Improvement of D* algorithm for Mobile Robot Path Planning in Partial Unknown Environment," Second International Conference on Intelligent Computation Technology and Automation, 2009.

[10] A. Stentz, "Focused D* Algorithm for Real-Time Replanning," Proceedings of the International Joint Conference on Artificial Intelligence, 1995.

[11] Hart, P., Nilsson, N., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cyber-netics, pp. 100,107, 1968.