IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719
Vol. 3, Issue 1 (Jan. 2013), ||V1|| PP 65-69

Autonomous robot motion path planning using shortest path planning algorithms

T. Ravi teja[1], S. Krishna chaitanya[2]
[1]Department of Mechanical engineering, Narasaraopeta engineering college, JNTU Kakinada
[2]Department of Mechanical engineering, St.marry's group of institutions, JNTU Kakinada

Abstract : Motion planning[1] is a fundamental problem in robotics and also arises in other applications including navigation, simultaneous localization and mapping (SLAM)[2], animation etc., in the motion planning normally using many techniques and algorithms. In this paper for path planning using a novel path planning algorithm called as Dijkstra's iteration (D*) inspired by the witkowski's algorithm based on a two dimensional grid map of the Static environment designed in mat lab using bioinformatics tools.

Keywords **-** Graph search, Mat lab, Mobile robotics, path planning, shortest path,

I. INTRODUCTION

In recent years, activity in robotics has swelled. Technological advances such as dramatically increasing computational power, memory size and communication bandwidth have lessened some of the old challenges of robotics. Furthermore, the community has more and more experience with the hardware challenges of building reliable robust platforms that can run for extended periods of time even in ever harsher environments. There exists also a push by funding agencies to produce practical robotics that run without human assistance in outdoor unknown environments, with sensing based on vision, and with the task of operating in a complex active environment with long-term global objectives.

II. MOTION PLAN CONTROL

Autonomous robot navigation[3] has long been a goal of researchers for applications ranging from military supply convoys, to space exploration, to autonomous highway driving. A critical requirement of higher level navigation applications is that the robot has some reasonable knowledge of its current position with respect to a fixed reference frame. For example, in navigation applications that entail motion to a target position, the robot needs an accurate estimate of its current position to plan a path to the goal and to conform success. Similarly, for exploration applications, position information can be used and recorded to avoid redundant coverage. The process of position estimation with respect to a fixed reference frame is defined as the localization of the robot.

A mobile robot can localize itself using two different classes of on-board sensors: pro-prioceptive sensors and exteroceptive sensors. Proprioceptive sensors, such as encoders or inertial measurement units (IMUs), measure the motion of the robot, acquiring data that can be integrated to estimate relative robot displacement. This method of localization is called odometry, or dead reckoning, and when used alone, the integrated error in global position grows without bound over time.

Exteroceptive sensors, such as laser range scanners or cameras, take measurements from the external environment. This data can be correlated at subsequent robot positions to compute relative pose or displacement estimates, which can improve and sometimes replace odometry. Externally sensed data may also be correlated with data from a global map, giving a global position measurement and bounding the overall position error. If a global map is not initially available, it is possible for the robot to build a global map with the externally sensed data, while using this map to localize. This approach is commonly called simultaneous localization and mapping (SLAM).

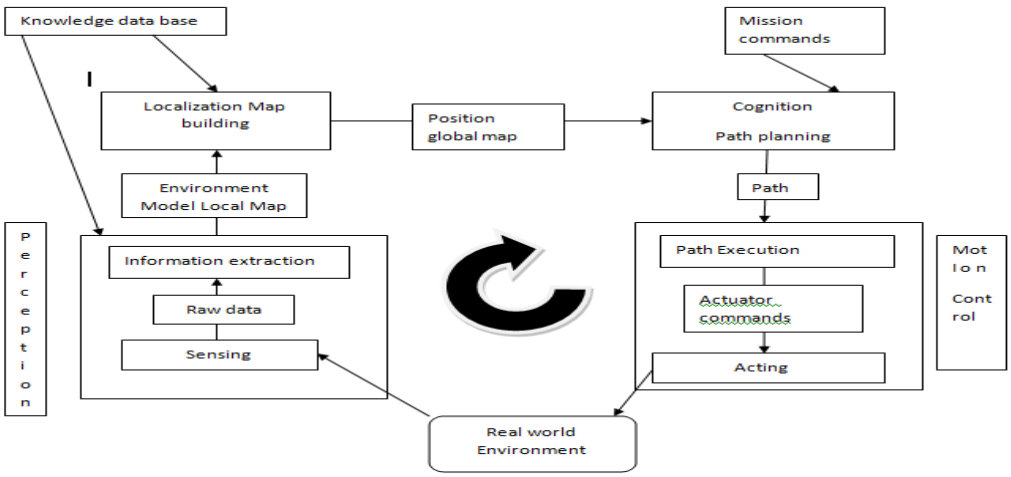

Fig 1. Architecture of motion planning control

The path planning problem is typically approached using a method of one of three categories: search-based, sampling-based, or combinatorial. By far, search-based methods currently dominate path planning in field robotics. The popularity of this method can be attributed first to the relative ease of its implementation and second to the early establishment of dynamic search-based algorithms. The sampling-based methods are relatively new to robotics and are the subject of a rush of recent research. These methods are particularly useful for planning of serial manipulators and in other situations where higher-dimensional planning is required. The combinatorial methods are the oldest and perhaps most studied branch of planning, having applicability in many areas, ranging from computer graphics to VLSI design.

(a) Search based (b) Sampling based (c) Combinatorial

Fig 2 : Types of planning

III. SHORTEST PATH PLANNING

Dijkstra's algorithm:

Dijkstra's algorithm[4] can be seen as an extension of BFS to allow for different edge costs it works by maintaining a cost function g (v) which is an estimate cost from the starting vertex to the vertex v. The function is initialized as zero for the start vertex and infinity for all other vertices. Vertices are visited in the ascending order of their g values. When a vertex is visited, it is marked as visited and the g values of all its unvisited neighbors are updated if the g value of the vertex plus the edge cost to the neighbor[5] is less than the g value of the neighbor. Once a vertex is marked as visited, its g value contains the exact cost from the starting vertex to that vertex. If the goal vertex is visited then a path is found, otherwise, if there are no more vertices to visit then there is no path. The actual path[6] can be obtained by maintaining back pointers which indicate for each vertex the vertex that expanded it.

1. **Algorithm** ***DijkstraDistances*** (***G, s***)[7]

```
1.  Q ← new heap-based priority queue
2.  for all v ∈ G.vertices()
3.  if v = s
4.  setDistance(v, 0)
5.  else
6.  setDistance(v, ∞)
7.  l ← Q.insert(getDistance(v), v)
8.  setLocator(v,l)
9.  while ¬Q.isEmpty()
10. u ← Q.removeMin()
11. for all e ∈ G.incidentEdges(u)
12. { relax edge e }
13. z ← G.opposite(u,e)
14. r ← getDistance(u) + weight(e)
15. if r < getDistance(z)
16. setDistance(z,r)
17. Q.replaceKey(getLocator(z),r)
```

IV. GRAPHS AND RESULTS

Mat lab program of Dijkstra's algorithm of using Bio-informatics tool box:

1. Create and view a directed graph with 6 nodes and 11 edges.

```
W = [.41 .99 .51 .32 .15 .45 .38 .32 .36 .29 .21];
DG = sparse([6 1 2 2 3 4 4 5 5 6 1],[2 6 3 5 4 1 6 3 4 3 5],W)
DG =        (4,1)      0.4500
            (6,2)      0.4100
            (2,3)      0.5100
            (5,3)      0.3200
            (6,3)      0.2900
            (3,4)      0.1500
            (5,4)      0.3600
```

```
(1,5)      0.2100
(2,5)      0.3200
(1,6)      0.9900
(4,6)      0.3800

h = view(biograph(DG,,'ShowWeights','on'))
[Dist, path, pred] = graphshortestpath (DG, 1, 6)
Dist =     0.9500
path =     1     5     4     6
pred =     0     6     5     5     1     4
set(h.Nodes(path),'Color',[1 0.4 0.4])
edges = getedgesbynodeid(h,get(h.Nodes(path),'ID'));
set(edges,'LineColor',[1 0 0])
set(edges,'LineWidth',1.5)
```

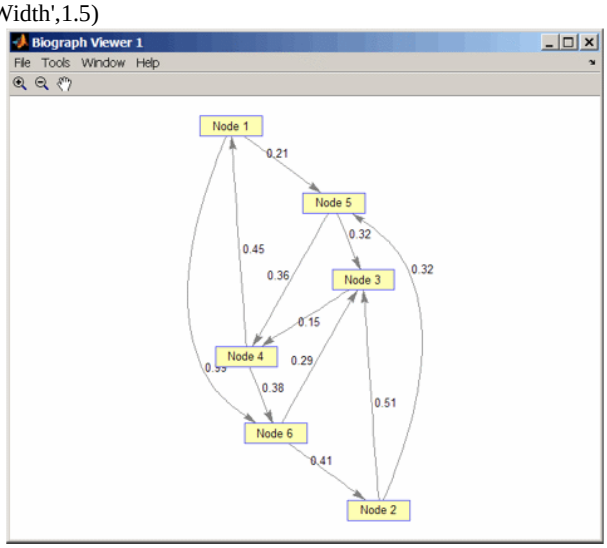

Fig 3: Biograph object with 6 nodes and 11 edges.

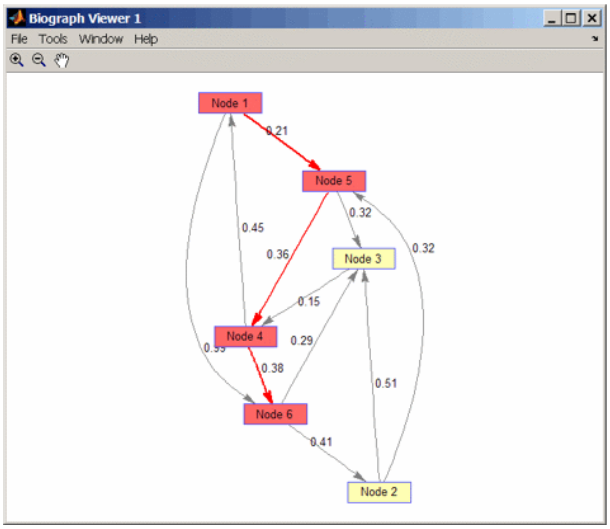

Fig 4. Mark the nodes and edges of the shortest path by coloring them red and increasing the line width

2. Create and view an undirected graph with 6 nodes and 11 edges.

```
UG = tril(DG + DG')
UG =    (4,1)      0.4500
          (5,1)      0.2100
          (6,1)      0.9900
          (3,2)      0.5100
          (5,2)      0.3200
          (6,2)      0.4100
          (4,3)      0.1500
          (5,3)      0.3200
```

```
            (6,3)       0.2900
            (5,4)       0.3600
            (6,4)       0.3800
h = view(biograph(UG,[],'ShowArrows','off','ShowWeights','on'))
[dist,path,pred] = graphshortestpath(UG,1,6,'directed',false)
dist =        0.8200
path =        1     5     3     6
pred =        0     5     5     1     1     3
set(h.Nodes(path),'Color',[1 0.4 0.4])
 fowEdges = getedgesbynodeid(h,get(h.Nodes(path),'ID'));
 revEdges = getedgesbynodeid(h,get(h.Nodes(fliplr(path)),'ID'));
 edges = [fowEdges;revEdges];
 set(edges,'LineColor',[1 0 0])
 set(edges,'LineWidth',1.5)
```

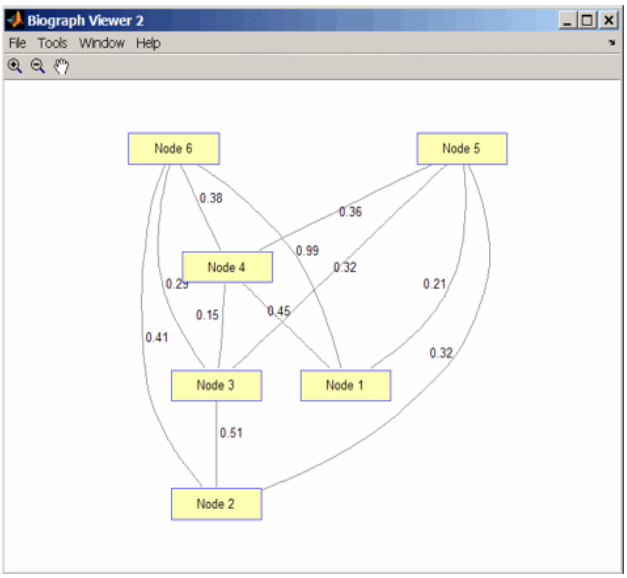

Fig 5: Biograph object with 6 nodes and 11 edges graph

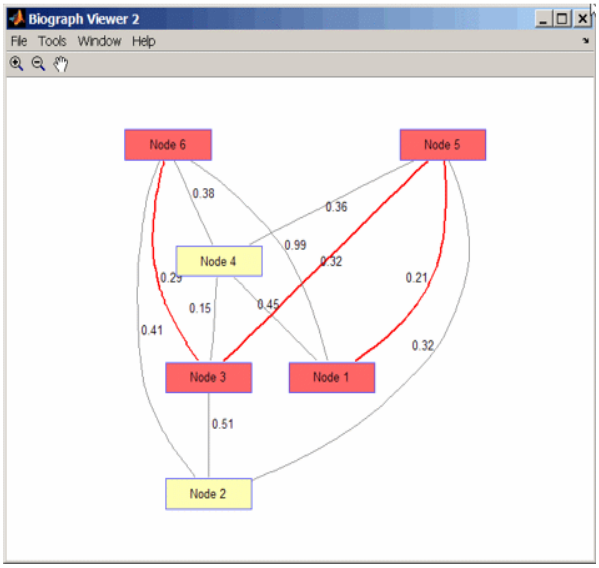

Fig 6. Mark the nodes and edges of the of undirect shortest path by coloring them red and increasing the line width

V. CONCLUSION

The new path planning algorithm is called D*[8] is proposed that finds optimal paths in weighted graphs, in occupancy grid maps with a safety cost mask around that obstacles. The path consists of straight line segments with continuous headings and is the shortest possible[9] path in geometrical space. Similarly to the witkowski's algorithm, which was used as an inspiration, the D* algorithm searches the graph forward and backward. However, unlike the witkowski's algorithm, which finds the optimal path only in binary occupancy grid maps, the D* algorithm also finds optimal paths in

weighted occupancy grid maps. The proposed d^* algorithm is tested and compared to the standard D^* [10]and witkowski's algorithm by simulation and experimentally under the same conditions in bio informatics tool box used in mat lab software[11]. The results of the tests confirm the expected advantages of the proposed path panning algorithm.

REFERENCES

[1]. J. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Dodrecht, Netherlands, 1991.

[2]. H. Choset and K. Nagatani. *Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization*. IEEE Transactions on Robotics and Automation, April 2001.

[3]. J. Crowley. *Navigation for an intelligent mobile robot*. IEEE Journal of Robotics and Automation, March 1985.

[4]. D. Ferguson and A. Stentz. *Field D*: An interpolation-based path planner and replanner*. In Proceedings of the International Symposium on Robotics Research, October 2005.

[5]. Felipe Haro and Miguel Torres. *A comparison of path planning algorithms for omnidirectional robots in dynamic environments.* Robotics Symposium, 2006. LARS '06. IEEE3rd Latin American, October 2006.

[6]. Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. Correction to "*A formal basis for the heuristic determination of minimum cost paths*". SIGART Newsletter, December 1972.

[7]. S. Koenig and M. Likhachev. *D* Lite. In AAAI Conference of Artificial Intelligence*, 2002.

[8]. S. LaValle. Planning Algorithms. The Cambridge University Press, 2006.

[9]. A. Stentz. *The focused D* algorithm for real-time replanning*. In Proceedings of the International Joint Conference on Artificial Intelligence, August 1995.

[10]. S. Koenig, M. Likhachev, *Fast replanning for navigation in unknown terrain*, IEEE Transactions on Robotics 21 (3) (2005) 354–363.

[11]. D. Ferguson, A. Stentz, *Field D*: an interpolation-based path planner and replanner, Robotics Research*, in: Results of the 12th International Symposium, ISRR, STAR: Springer Tracts in Advanced Robotics Series Volume 28, 2007,pp. 239–253.