The 5th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2008)

Time-Suboptimal Real Time Path Planner for a Humanoid Robot Under Linear Motion Constraint

Soo-Hyun Ryu, Korea Univ., okrsh@korea.ac.kr
Young-Hoon Lee, Korea Univ., younghun2@korea.ac.kr
Yeonsik Kang, KIST, yeonsikkang@gmail.com
Bum-Jae You, KIST, ybj@kist.re.kr
Nakju Lett Doh, Korea Univ., nakju@korea.ac.kr

Abstract – The purpose of this paper is to plan a path for humanoid robot called MAHRU on real-time in a partially dynamic environment. And a path planner should consider the kinematic constraints of the humanoid robot and generate a line-based and time-suboptimal path. For these purposes, we take a visibility graph approach and a global-local hybrid approach. By using a topological map and door information, we can get the fault-tolerant path and obtain each local path planner's input information. Moreover, by reducing the number of turning points, the path planner generates a time-suboptimal path. The results of simulation present that a calculation time of the global-local method is 6 times less than that of the local method and a moving time of the time-based visibility graph is 1.5 times less than that of the previous visibility graph.

Keywords - Humanoid robot, Real-time, Global-local, Visibility graph, Time-suboptimal

1. Introduction

In this paper, we develop a path planner for a humanoid robot called MAHRU [1] as shown in Fig.1. Our design purposes are as follows: an adequate path planner for a partially dynamic environment, path planning on real time, and to obtain kinematics-based path, line-based path and time-suboptimal path.

To be an adequate path planner for a partially dynamic environment, we classify the dynamics into two veins. One is a constrained dynamics which moves under pre-defined constraints such as doors, and the other is an unconstrained dynamics which happens arbitrarily such as human passerby. We develop a path planner for an environment with the constrained dynamics.

Real time path planning means that a path planner can generate path on the fly. And the path planner considers the kinematic constraints of the humanoid robot for obtaining the kinematic-based path. The path is generated by sequences of combination of forward, backward and side step [2]. Also, the robot performs a pure rotation by encircling an arc as shown in Fig.2.

Fig.1. MAHRU

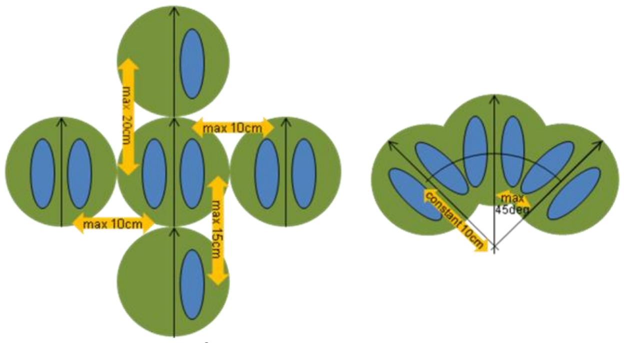


Fig.2. Information about MAHRU's step

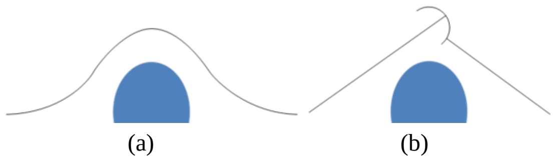


Fig.3. Designed paths (a) Smooth path (b) Line-based path

Different from mobile robots, the humanoid robot cannot change its heading angle abruptly. For example, it is difficult for the humanoid robot to track a smooth path as shown in Fig.3(a). We believed that, for the sakes of stability and the control, it is better to design a line-based path as shown in Fig.3(b) than the smooth path. For a line-based path, turning tasks need large amounts of moving time because the robot should decelerate first and accelerate again in that point. Thus, in the view point of moving time, a path with minimum turning point is more adequate than a path with the shortest distance as shown in Fig.4. In this paper, we seek for a path with minimum moving time. Yet, our approach is based on a heuristic so we call this path the time-suboptimal path.

For the real time calculation, we adopt a global-local hybrid approach [3]. We divide the map into various cells and construct a topological map that consists of nodes and edges. For a given start and end point, we generate sequences of nodes using Dijkstra algorithm [4] and call this process a global path planning. For each cell in the global path, we utilize a local path planner. There are

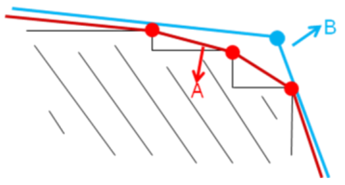


Fig.4. A shortest path A and a minimum time path B. Moving distance of the path A is shorter than the path B, but, moving time of the path A is longer than the path B because of turning time of a robot.

various successful approaches such as the potential field method (PFM) [5], the A* algorithm [6] and the visibility graph method [7] for a local path planner. But these methods have advantages and disadvantages to be applied to the local path planner. The PFM is usually used in real-time path planning, but the PFM does not generate the line based path. The A* algorithm used in real time path planning generates the line-based path but the algorithm only works in a discrete space. On the other hand, the visibility graph can generate line-based path and works in a continuous space. Thus, we choose the visibility graph method and extend the previous visibility graph to time-based visibility graph and then reduce the number of turning points to find a time-suboptimal path.

There are three contributions in this paper. The first one is that we designed a global path planner which can be used in a partially dynamic environment with door states changing. When some of doors are closed and there is no solution in the topological map, our planner yield a path that the end node is the closest node from the unreachable destination. The second contribution is that our local path planner generates the time-suboptimal path by reducing the number of turning points. The third one is that, by virtues of the aforementioned two properties, we designed a path planner that satisfies our design purposes.

We introduce the global path planner that enables us to deal with doors and to plan the fault-tolerant path in section 2. In section 3, we introduce how to concern various kinds of steps and take the time-suboptimal path as local path. Finally, we present the simulation results of comparison between the previous algorithm and the proposed algorithm using Matlab in section 4.

2. Global Path Planner

2.1 Global Path Planning

In indoor environment, there are doors and their narrow ways for separating each area. These can be characteristics of an indoor environment, but at the same time these are critical obstacles for a robot. Especially, humanoid robots have a big shakes in a lateral direction and difficulties to track a path. Because of these reasons, when a humanoid robot passes a door area, we have to pay more attention than the other situations. For this defect, we propose a method that includes the door information on a database and utilizes them. The door information includes two points which a robot passes through safely and a state of the door, whether it is open or closed. And this information

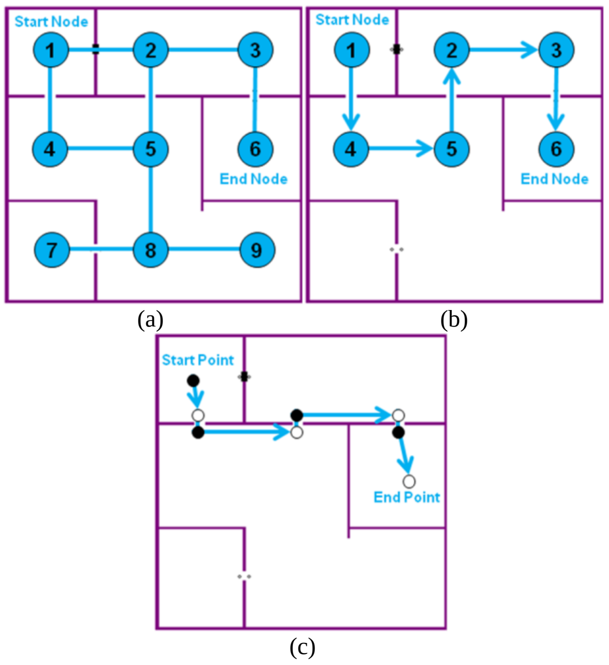


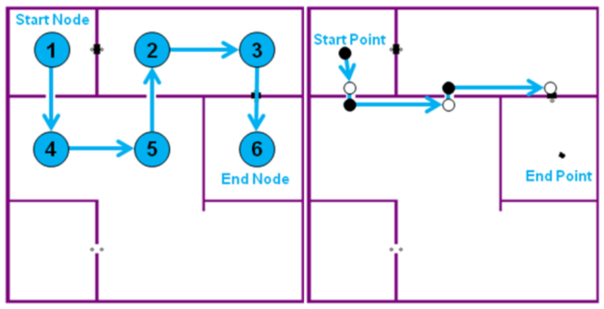
Fig.5. Global path planning: (a) It presents information of a start position, an end position, a door and a topological map. We hope to get the path from the node no.1 to the node no.6, and the door between them is closed. (b) It shows the path of the global path planning. (c) The generated path by the global path planner is separated four node groups -①, ④⑤, ②③, ⑥- by doors and each node group executes a local path planner. In this figure, the black points are used as a start point by each local path planning and the white points are used as an end point.

is contained in a connect information between each local area.

For this method, when a robot moves from a local area to the other area as shown in Fig.5, the global path planner can generate the path passing through the two points of door by using the database. Accordingly, the planner generates a safe and suitable path on changes of a door state. When passing through a door two points, one white point and one black point in each area as shown in Fig.5(c), across a door are created. Every black point is assumed to be a starting point and every white point is assumed to be a end point in each local path planning.

2.2 Fault Tolerant Path Planning

If a path is blocked by obstacles, a general path planner cannot solve this problem or generates a unreasonable path because it treats the dynamic obstacle as a static obstacle or no obstacle. Accordingly, in the situation shown in Fig.6, conventional path planners cannot generate any path. For this reason, we propose an algorithm that generates the shortest path as close to the end position as possible -passing through a minimum number of dynamic obstacles- regardless of the path blocked by obstacles. The Dijkstra algorithm uses open list and closed list when searching a path in implementation. On the other hand, the proposal algorithm uses an additional dynamic list. In the process of searching paths, the algorithm stores new searched nodes in the open list, already used nodes or unreachable nodes in closed list and new searched nodes but dynamic obstacle nodes in dynamic list. In other words,



(a) (b)

Fig.6. Fault tolerant path planning: (a) It presents the path of the global path planning. It shows that the global path planner generates the path passing through between the node no.3 and the node no.6 in spite of blocking the path by the closed door. (b) The generated path by the global path planner is separated four node groups -①, ④⑤, ②③, ⑥- by doors. The rest node groups except the group included the node no.6 execute local path planner because of the door between the node no.3 and the node no.6.

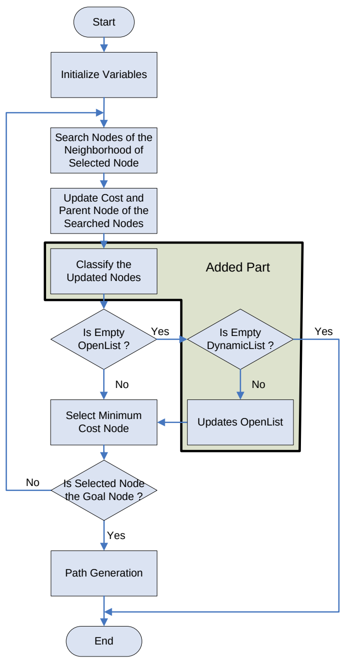
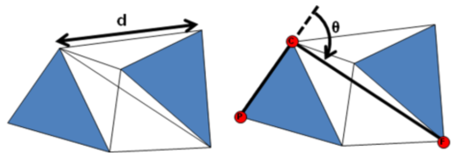

Start
Initialize Variables
Search Nodes of the Neighborhood of Selected Node
Update Cost and Parent Node of the Searched Nodes
Classify the Updated Nodes
Added Part
Is Empty OpenList ?
Yes
Is Empty DynamicList ?
Yes
No
No
Select Minimum Cost Node
Updates OpenList
No
Is Selected Node the Goal Node ?
Yes
Path Generation
End

Fig.7. Fault tolerant algorithm flow chart

a node needed to be reviewed is stored in the dynamic list. In the process of searching area, if open list is empty, the algorithm moves the stored nodes in the dynamic list to the open list for a dynamic obstacle node to be a basis node and then initializes the open list. By this method, we can get the shortest path that reaches the end position while passing through a minimum number of dynamic obstacles on the same time complexity as the existing Dijkstra algorithm.

3. Local Path Planner

For a time-based path planning, we utilize a method that reduces the number of turns after planning a path based on a visibility graph. At this time, each node position of the visibility graph is virtual turning point. We regard a sum of one continuous turn and one continuous step as one



(a) (b)

Fig.8. Information of a distance and an angle included in the visibility graph: (a) Distance information between arbitrary two nodes. (b) Angle information between arbitrary three nodes. This presents the angle which a robot should turn when it moves from the past node to the future node passing the current node.

command, and then the path can be presented by the sum of commands.

3.1 Time-based Visibility Graph

The existing visibility graph contains distances between each node. In this paper, we add angles among three nodes on the visibility graph. We regard arbitrary three nodes as a past node, a current node and a future node and then we can calculate an angle of these nodes by the method shown in Fig.8(b). Table 1 shows the maximum angle, the maximum distance by a step type and the time required for a turn or step. Accordingly, we can calculate the time of each command by using the information of the distance between the current node and the future node and the angle configured by the past node, the current node and the future node. By using the calculated time information, the local path planner can generate the time-based path. But, using the time value which of each consists of three nodes cause the data size to be increased 2-dimension to 3-dimension and the time complexity to be increased like that of data size.

Table 1. Information about MAHRU's step

	Forward	Backward	Side	Turn
Moving Distance (or Turning angle)	Max 20cm	Max 15cm	Max 10cm	45deg
Moving Time	1.4sec	1.4sec	2.8sec	2.5sec

3.2 Angle Expansion on Various Step Types

The time-based path concerned of step types can be generated by expansion of various step types. MAHRU can step 4 directions, forward, backward, left side and right side step. And the turning angles depend on kinds of past step from past node to current node and future step from current node to future node. According to this

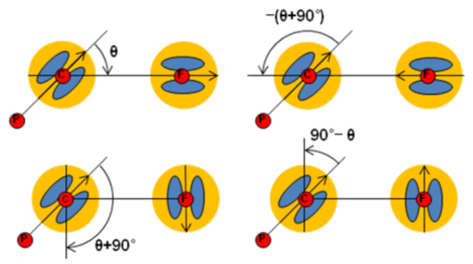
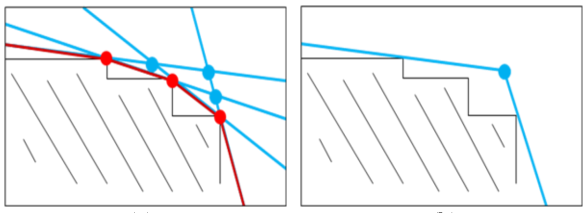
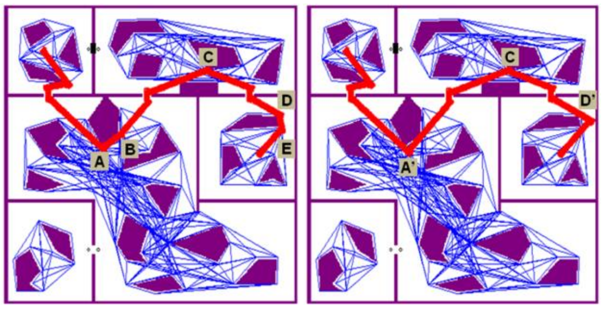


Fig.9. Angle information followed by step types: This presents the angle by a step type of moving from current node to future node when a robot moved from past node to current node by forward step.



(a) (b)

Fig.10. Reducing the number of turning points: (a) Red segments are generated path by time-based visibility graph and blue straight lines are the expansion of red segments and each point is the intersection of lines. (b) Path of minimum turn frequency.



(a) (b)

Fig.11. Local path planning: (a) This presents visibility graph generated by time-based visibility graph. The point C shows that the path is not shortest path. (b) This presents the path reducing the number of turns. These figures show the change from point A, B to point A' and point D, E to point D'.

method, data size of angle information is expanded the square times of number of step types, but this method can plan the path at the same level of time complexity as the path planning without considering different types of steps.

3.3 Reducing the Number of Turn Points

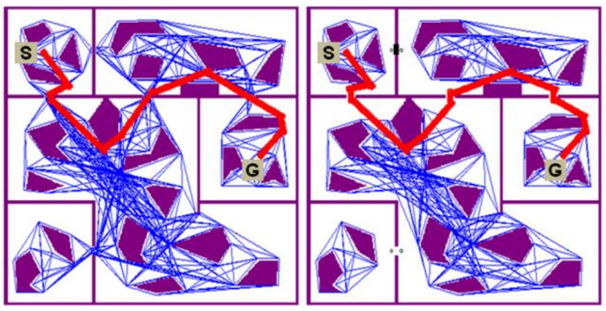
The generated path by the time-based visibility graph is not the minimum time path because nodes of the visibility graph are not candidate nodes of minimum time path but that of the shortest distance path. For generating the minimum time path, we extend each segment of the visibility graph to the line and regard intersecting point of two lines as a node. At this time, if every point of intersection of all lines is considered, not only data size but also time complexity will increase in geometric progression. To avoid problem, we only extend the segments of the generated path by time-based visibility graph to the line and find their intersection and calculate moving time on them. After that, we compare the time with previous time and then choose the minimum time path among searched paths. With this method, we can take the path which is not the shortest one but the path that take the least time.

4. Simulation Result

4.1 Comparison between Global-Local Method and Local Method

Table 2. Comparison between calculating times

	Time-based Path	Shortest Path
Local Method	18.36 sec	2.07 sec
Global-Local Method	2.77 sec	0.51 sec



(a) (b)

Fig.12. Paths of local method and global-local method: (a) The generated path by local path planning on whole area. (b) The generated path by global-local method proposed in this paper.

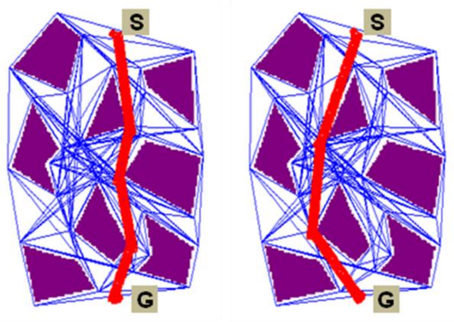
We have much better performance as shown in Table 2 by using the global-local method. Furthermore, the path shown in Fig.12(a) passes through narrow door area diagonally while the path shown in Fig.12(b) does not pass through the area diagonally. Therefore, the generated path by the global-local method is safer than the other paths. And local method dose not generate the path blocked by closed doors. In other words, global-local path planner can generate safer and fault-tolerant path by using door information.

4.2 Comparison between an Existing Visibility Graph and a Time-based Visibility Graph

Table 3. Comparison between shortest path and time-based path

	Visibility Graph	Time –based Visibility Graph
Moving Distance	14.75 m	15.63 m
Moving Time	148 sec	101 sec
Time Complexity	$O(N^2)$	$O(N^3)$

Comparing between the shortest path and the time-based path, a moving distance of the time-based path is not much longer than that of the shortest path but a time of the time-based path is much less than that of the shortest path, as shown in Fig.13. On the other hand, the time-based visibility graph has a disadvantage that time complexity and data size are larger than those of existing visibility graph.



(a) (b)

Fig.13. Paths of an existing visibility graph and time-based visibility graph: (a) Shortest path. (b) Time-based path.

5. Conclusion

This paper contributes that we designed a global path planner which can be used in a partially dynamic environment with door states changing. Although there is no solution because of some of doors closed in the topological map, the proposed planner yields a path that leads the robot to the closest node from the unreachable destination. And the second contribution is that the proposed local path planner generates the time-suboptimal path by reducing the number of turning points. But it has some defects as well which will be studied more about making configuration space and linear angle.

References

[1] B.-J. You, D. Kim, C. Kim, Y.-H. Oh, M.-H. Jeong and S.-R. Oh, "Network-based Humanoid 'MAHRU' as Ubiquitous Robotic Companion," in Proc. of the Int. Fed. of Automatic Control, pp.724-729, 2008.

[2] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "Real-time path planning for humanoid robot navigation," *in Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI'05),* pp. 1232-237, 2005.

[3] J. S. Zelek and M. D. Levine, "Local-global concurrent path planning and execution," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 30, no. 6, pp. 865-870, 2000.

[4] S. M. LaValle, *Planning algorithms.* Cambridge, U.K.: Cambridge Univ. Press [Online]. Available http://msl.cs.uiuc.edu/planning/, 2006.

[5] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE Int. Conf. Robot. Autom*., pp. 1398-1404, 1991.

[6] C. W. Warren, "Fast path planning using modified A* method," in *Proc. IEEE Int. Conf. Robotics Automation*, pp. 662-667, 1993.

[7] Huang H.-P., S.-Y. Chung, "Dynamic Visibility Graph for Path Planning," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2813-2818, 2004.