A heuristic Sweeping Algorithm for Autonomous Smearing Robot

W.K.Hyun*

*Dept. Electronics Eng. of Honam Univ, KwangJu, KOREA

E-mail: wkhyun@honam.honam.ac.kr

Abstract

A heuristic sweeping algorithm for an autonomous smearing robot which executes the area filling task is proposed. This algorithm searches tracking points heuristically at the contact position with the obstacle and environment wall while the robot tracking whole workspace, and finds sequential tracking line by sequentally connecting the tracking points in such a way that (1) the line should be never crossed, (2) the total tracking points should be is linked as short as possible, and (3) the tracking link should be cross over the obstacle in the work-space. If the line pass through the obstacle, hierarchical collision free algorithm proposed is implied. The proposed algorithm consists of (1) collision detection procedure, (2) obstacle map making procedures, (3) tracking points generation procedures for subgosls, (4) tracking points scanning procedures, and (5) obstacle avoidance procedure.

1. Introduction

The development of specialized service robots was brought about by the increasing demand to economize on several kinds of services. A representative application fields are floor-cleaning or smearing of construction. Most of today's cleaning and smearing machines are guided by human operators along extended and fixed motion patterns in areas often greater than some hundreds of square meters. Due to increasing machine maintenance and personal costs this cleaning or smearing task could be performed in future by an autonomous robot. Many researcher have been devoted to the path planning algorithm[1,2,3,4,5]. In these methods, the methods in [1,2,3] developed semi-autonomous sweeping robot and applied it to the floor-cleaning tasks. These robots may meet dead-lock station in some environment with obstacles owing to local collision avoidance approach based on sensors. Hofner proposed a heuristic path planning algorithm using potential field function[5] which is bounded in local planning approach. This method also can not solve the dead-locking problem.

In this paper, a path planning algorithm for an autonomous sweeping robot which executes the area filling task is proposed. This algorithm generates subgoals at the contact position with the obstacle and environment wall while the robot tracking whole workspace, and finds sequential tracking line which link the subgoals continuously and is never crossed. If the line pass through the obstacle, hierarchical collision free algorithm proposed in [6] is implied.

2. A Heuristic Sweeping Algorithm

Since a sweeping robot works for floor-cleaning or smearing in 2 dimensional space, a mobile robot and objects in workspace are modeled in 2 dimensional space. It is assumed that (1) the mobile robot move in omni-direction, (2) the robot is a polygon shape and obstacles in workspace have polygon or circle shape.

Area filling algorithm consists of obstacle map making procedure, and tracking point generation procedure, tracking point scanning procedure, and robot language generation procedure. Fig. 1 represents the sequential procedure of the proposed area filling algorithm. The sweeping robot can have translational and rotational motion in 2 dimensional work space. Therefore the configuration space of the robot in which the robot is described as a point is 3 dimensional space. This configuration space is called as obstacle map.

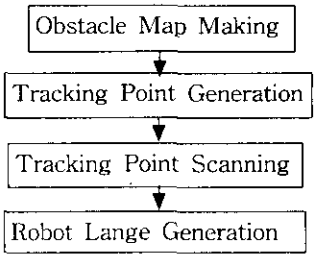

Fig. 1 The sequential procedure of area filling Algorithm

The obstacle map making procedure is to make the obstacle map by scanning the whole work space. And then the configuration space is quantized as a equal size cell. A cell in the grid-based configuration space is identified as FREE cell or OBSTACLE cell. The grid-based configuration space is utilized for finding a collision free path by using Two Level path planning algorithm proposed in [6] which is a kind of graph search algorithm. Tracking point generation procedure tracks the whole workspace with an adjustable scanning width and records tracking points as subgoals and then the path planner shift to the tracking point scanning procedure. This procedure link the recorded tracking points continuously under the condition that a tracked point is never tracked and the linked path is never crossed. The next procedure is robot motion generation procedure.

2.1 Tracking point generation procedure

Tracking point generation procedure tracks the whole workspace with an adjustable scanning width and records tracking points as subgoals. In this tracking, the separation of each track is set to be half of the width of the robot. The separation can be varied and with smaller separation of the tracks, the degree of overlap increases. If, for example, the width is the same as the width of the robot, then there is no overlap between the tracking lines. While the robot tracks the whole workspace, tracking point generation procedure records the contact positions and collision status between robot and wall of workspace, and robot and obstacles. These positions are utilized as subgoals to take area filling. The data structure of tracking points is as follows.

```
struct trackPoint {
                int   position[DIMENSION];
                char  status;
                char  tracked_flag;
                int   obstacleNo;
} *TrackingPoint;
```

Where position[DIMENSION] is the contact position between robot and wall and robot obstacle, and status represents contact types which are LEFT_WALL, RIGHT_WALL, LEFT_OBSTACLE, and RIGHT_OBSTACLE. Each means contact state between left side of robot and wall, right side of robot and wall, left side of robot and obstacle, and right side of robot and obstacle, respectively. And tracked_flag implies that tracking point is tracked or not. It will be utilized when the robot tracks these tracking points for finding area filling path. In tracking point generation processor, the robot moves from left top to right bottom in workspace to record the tracking points and the orientation of the robot is fixed as 0 degree. The algorithm of tracking points generation is as follows;

For given robot initial position $(x, y, \theta) = (0, 0, 0^0)$, and scanning width Δy and Δx,

x and y mean positions along x axis and y axis, respectively.

STEP 1: Set x and y to 0, respectively, And find a initial robot position in the workspace by increasing x with Δx and y with Δy, respectively, while orientation of the robot if fixed as 0 degree. And record its position and status as LEFT_WALL.

STEP 2: Move the robot from left to right with y position fixed. If the robot contacts the wall, record its position and status, where the status is RIGHT_WALL. Else if the robot collides with obstacle, record its position and status as RIGHT_OBSTACLE and obstacle number. Move the robot until it meets right wall.

STEP 3: If the robot meets right bottom of the workspace, goto STEP 5. Move the robot as

Δy along y axis while x position is fixed. And then move the robot from right wall to left wall. If the robot collides with obstacle, record its position at the contact position and recode the status which are LEFT_OBSTACLE and obstacle number. And move the robot until the robot meets left wall. And record the status and position.

STEP 4: Move the robot as Δy along y axis and goto STEP 1.

STEP 5: Sort the recoded data.

2.2 Tracking point scanning procedure

Tracking point scanning procedure links the

recorded tracking points continuously under the condition that a tracked point is never tracked and the linked path is never crossed. The next procedure is robot motion generation procedure.

Tracking point scanning procedure find a continuously linked line by connecting all tracking points in such a way that (1) a tracked point should not be tracked, and (2) the integrated length of linked line should be as short as possible. Tracking point scanning procedure find a tracking point to be next linked with respect to the status of the current tracking point. The tracking strategy is as follows;

For given scanning width , and minimum length of the workspace along y ax $\Delta y$is, y_{max}.

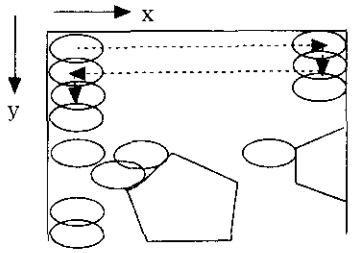

Fig. 3 An example of tracking point scanning

Step 1: Search TP's(Tracking Points) of RIGHT_WALL or RIGHT_OBSTACLE TP

Step 2: Find minimum distance TP from current TP in Step 1, and if it $y=0$is not tracked and not collided with obstacle when current moves to the TP goto Step 6 else goto Step 3

Step 3: $y \leftarrow y - \Delta y$, if goto Step 5

Step 4: Search the same status, RIGHT_WALL and RIGHT_OBSTACLE TP's, and find minimum distance and untracked one. If there is no one, goto the next Step else goto Step 6.

Step 5: $y \leftarrow y + \Delta y$, if $y > y_{max}$, then goto Step 6

Step 6: Find next TP to be linked which is minimum distance one between current TP and the next TP to be linked in whole untracked TP's. In this case collision free path planning procedure is requested.

Step 7: If the number of the linked tracked points is the same as tracking points stored in tracking point generation procedure, then end, else goto link the TP and mark the TP tracked.

(2) When the status of the current tracking point is RIGHT_WALL:
the process is the same as the case when the current TP is LEFT_WALL except that in Step 1 and Step 4, LEFT_WALL and LEFT_OBSTACLE are replaced with LEFT_WALL and LEFT_OBSTACLE, respectively.

(3) When the status of the current tracking point is LEFT_OBSTACLE:
the process is the same as the case when the current TP is LEFT_WALL except that in Step 1 and Step 4, RIGHT_WALL and RIGHT_ OBSTACLE are replaced with LEFT_WALL and LEFT_OBSTACLE, respectively, all candidate TP should be the same obstacle number state.

(4) When the status of the current tracking point is RIGHT_OBSTACLE:
the process is the same as the case when the current TP is LEFT_OBSTACLE except that in Step 1 and Step 4, LEFT_WALL and LEFT_OBSTACLE are replaced with LEFT_WALL and LEFT_OBSTACLE, respectively.

2.3 Collision free path planning procedure

When tracking point scanning procedure can not search a TP to be linked near the current TP. It find a minimum distance TP from current TP in whole untracked TP. If the current TP collide with the minimum distance TP when current TP moves linearly to the minimum distance one, collision free path planning procedure in [6] is requested.

3. Simulation results

The area filling algorithm has been tested with mobile robot of octagon type. In test-bed example as shown in Fig. 5, there are 5 obstacles in workspace and it is the received draft data from GUI interface. The workspace is 50m×50m square meter. It is assumed that the robot can move in omni-direction. The results for sequential procedures of the proposed algorithm are shown in Fig. 6 through Fig. 9. Subgoals generated by tracking point generation procedure are shown in Fig. 6. Fig. 7 shows that the resulting path of tracking point scanning procedure without collision free path planning. The 5 collision situations are occurred in the test-bed example. Planned collision free path by TLPPA are illustrated for each collision situation in Fig. 8. Fig. 9 show that the proposed area filling can plan a successive path. The time cost for planning the final path is about 3 sec.

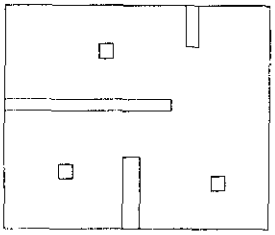

Fig. 5 An exemplary map

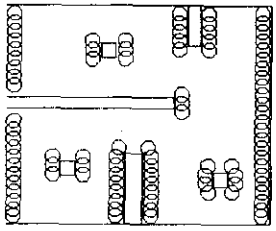

Fig. 6 Generated tracking points

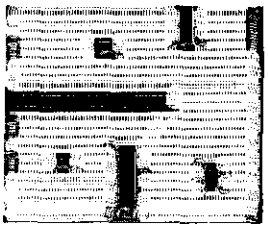

Fig.7 Area Filling without Obstacle avoidance

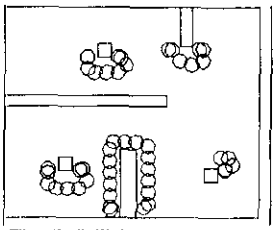

Fig. 8 Collision free path

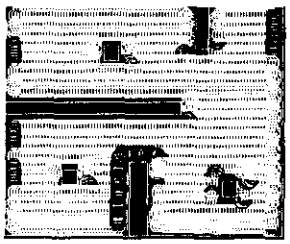

Fig. 9 Final Result

4. Concluding remarks

In this paper, A heurictic sweeping algorithm for cleaning robot is proposed. The proposed area filling algorithm can fill successfully whole workspace without deadlocking and collision in spite even though obstacles are in workspace. In the algorithm, tracking point generation procedure to find subgoal for filling path, and tracking scanning procedure to link the subgoals without discontinuity and collision for finding filling path are proposed. As a future work, the area filling path planning algorithm works in an environment with unknown obstacles will be investigated.

[Reference]

[1] F.Yasutomi, D.Takao, M.Yamada, K.Tsukamoto, "Celaning robot control," *proc. if IEEE International Conference on Robotics and Automation*, vol.3, pp1839-1841, 1988

[2] T.Fukuda, S.Ito, N.Oota, F.Arai, Y.Abe, K.Tanaka, Y.Tanaka, "Navigation system based on ceilling landmark recording for autonomous mobile robot," *proc. of international Conference on Industrial Electronics, Control, and Instrumen- tation*, vol. 3, pp.1466-1471, 1993

[3] W.Li,"Preception-action behavior control of a mobile robot in uncertain environments using fuzzy logic," *proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, vol.1, pp439-461, 1994

[4] D.Kurabayashi, J.Ota, T.Arai, and E.Yoshida ,"Cooperative Sweeping by Multiple Mobile Robots," *proc. of IEEE Conference on Robotica and Automation*, vol. 2, pp1744-1749, 1996

[5] C.Hofner and G.Schmidt,"Path planning and guidance techniques for an autonomous mobile cleaning robot," *proc. of IEEE Conference on Intelligent Robots and Systems*, vol. 1, pp610-617, 1994

[6] W.K.Hyun, "A Hierarchical Collision-Free Path Planning Algorithm for Robots," *IEEE IROS '95*, Vol.2. pp.837-845, 1995.