LETTER

Robot Exploration in a Dynamic Environment Using Hexagonal Grid Coverage

Kihong KIM", Nonmember and SeongOun HWANG"†, Member

SUMMARY Robot covering problem has gained attention as having the most promising applications in our real life. Previous spanning tree coverage algorithm addressed this problem well in a static environment, but not in a dynamic one. In this paper, we present and analyze our algorithm workable in a dynamic environment with less shadow areas.

key words: robotics, spanning tree coverage, exploration, hexagonal grid coverage

1. Introduction

The robot covering problem can be formulated as follows. Given a continuous work area, a robot moves along a path such that every area in the work area is covered along the path. Robot covering is one of very promising applications such as cleaning, lawn mowing, harvesting, search and rescue, and mine detection in our life.

Traditional coverage algorithms used by cleaning robots are based on random moving ([1],[2]). However, they have a shortcoming that they take a long time to complete their mission. To deal with this problem, a lot of researches have been made. One representative approach is based on spanning tree coverage (STC), where a robot decomposes a work area into complete square-shaped cells, makes a spanning tree, and covers the entire work area [3]. STC optimally solves the single-robot coverage problem. STC algorithms have also developed into MSTC (multi-robot spanning tree coverage) ([4],[5]) and MFC (multi-robot forest coverage) algorithms [6]. Existing STC algorithms have focused on two issues: full (a.k.a complete) and non-repetitive coverage. However, those approaches have the following disadvantages:

First, those may not be workable in a specific dynamic environment, where obstacles are unexpectedly added to the work area (for example, in the open air, stones or trees come down and block off the returning path) after a spanning tree was established (See Fig. 1 and Fig. 2).

Second, STC algorithms generate a large portion of shadow areas from the given work area. It is because in STC, each cell of size 2D is partitioned into four sub-cells of size D (See Fig. 3), and the areas denoted by X can be fallen into shadow area even though they are large enough to be passed by a robot (See Fig. 4).

Third, all online versions of those approaches require recursive calls whose number grows in linear with the total number of cells within the work area. This is very inappropriate in a mobile robot environment where a robot agent has a very limited memory space but the coverage area is very broad, particularly in outdoor environments.

Therefore, we need a full coverage algorithm workable in a dynamic environment which guarantees non-repetitive or minimal repetitive coverage. In this paper, we present and analyze the algorithm called HGC (hexagonal grid coverage).

2. The HGC Algorithm

In case of STC algorithms, there are off-line and on-line versions. The off-line version starts coverage with perfect a priori knowledge of its environment. In contrast, the on-line version starts coverage with partial knowledge of its environment. The knowledge is gradually incremented while covering the work area. HGC algorithm is basically an on-line version, but also works in off-line way.

In this paper, we make some assumptions as follows:
Table 1 HGC Algorithm.

| S = the starting cell;        | X = the current cell under coverage; |
|-------------------------------|--------------------------------------|
| P = the previously covered cell shortly before X; | N = the new neighbor cell(s) of X or the covered cell(s); |
| V = the number of adjacent cells which a cell has; | Y = the new neighbor cell with the least V, which is located nearest in counterclockwise order, starting with P; |
| B = the bridge cell(s); | |

HGC Algorithm:

Input: A starting cell S
Initialization: Call HGC(S, Null):

HGC(X, P)
{
    It covers X;
    while(X has uncovered neighbors)
    {
        Check if pre-stored B exists among the neighbors;
        If B exists, decrement the value V of B by 1;
        Store the locations of X’s new neighbors denoted as N on the memory;
        Check if the neighbors are bridge cells;
        If bridge cells exist, denote them as B whose V is set to 7;
        Calculate the value Vs of the neighbors which are not bridge cells;
        if(X == S) // The current cell is the starting cell:
            Denote B as Y, where B’s V is the least among Bs and located nearest in counterclockwise and forward-moving order;
        else // X has neighbors which are not B
            Denote the neighbor cell as Y, where its V is the least among the neighbors and located nearest in counterclockwise and forward-moving order;
        }
    else // The current cell is not the starting cell;
    if(X has only B around it)
        Denote B as Y, where B’s V is the least among Bs and located nearest in counterclockwise order, starting with P;
    else // X has neighbors which are not B
        Denote the neighbor cell as Y, where its V is the least among the neighbors and located nearest in counterclockwise order, starting with P;
    }
    It moves to Y;
    P is set to N, X is set to P, and Y is set to X;
}
with obstacles added. We assume that the robot currently covers \((19,-3)\) as in Fig. 8. At that time, an obstacle happened as in Fig. 9. Figure 10 shows that the robot covered up to \((21,5)\). The robot has three new neighbors, whose coordinates are \((20,4)\), \((19,5)\) and \((20,6)\). Among them, \((19,5)\) and \((20,6)\) become bridge cells. The cell at \((20,4)\) becomes \(Y\), which will be covered by the robot at subsequent stage. As in Fig. 11, the robot continues its covering until it reaches \((18,-2)\). Note that there is no \(Y\), but only B's exist around the robot. A number of B's form a single path, which the robot uses to cover the remaining work area as shown in Fig. 12. As in Fig. 13, the robot finally completes its coverage.
3. Algorithm Analysis

**Theorem 1** (Complete coverage): The HGC algorithm completely covers every cell which is accessible from the starting cell S.

**Proof:** The covering robot divides the work area into a set of cells and stores their locations. The work area consists of hexagonal cells which are identical in shape and size. Each cell has its adjacent cells, whose number ranges at least two through six (a leaf cell which has a single adjacent cell is excluded in the paper). Therefore, the component of the work area grid accessible from S is a connected graph. The robot can move from the current cell to one of its adjacent cells. There is no gap among cells. Therefore, the robot can cover all cells accessible from S.

**Theorem 2** (Non-repetitive coverage): The HGC algorithm does not repeatedly cover any point in the work area grid.

**Proof:** To prevent repetitive coverage, we introduced the concept of bridge cell. Bridge cell is only one path which connects a cell with its adjacent cells. The robot usually probes normal cells at first. The robot continues probing in one direction, and then finally reaches a deadend point. If there is no normal cell, then the robot checks if there is any bridge cell around it. A number of bridge cells form an exit way which allows the robot to get out of the deadend point. In addition, the robot stores the coordinates of the covered area on its memory. Therefore, the robot can cover any work area point without any repetition.

4. Comparison with STC

Figure 14 shows cell parameters for comparison of HGC with STC.

Figure 15 shows that the number of cells covered by HGC is greater than those by STC. It means that the size of shadow areas which the robot cannot cover is decreased in our algorithm. However, the coverage distance is increased a little bit. From simulations in various work areas, we also found that STC is suitable for square-shaped work area, but HGC for rectangle-shaped one. Work areas in real environment are closer to rectangle-shaped areas with a number of curves.

5. Future Work

Our future work is to address repetitive coverage in more complex work areas, which are not addressed in the current HGC algorithm and classified into two cases: The first case is, either leaf cells exist or single paths exist. The second case is, the work area gets more complex due to obstacles which happened after the robot starts its coverage. For example, a leaf cell can be generated due to obstacles happening in the way.

Acknowledgements

This work was supported by 2013 Hongik University Research Fund.

References

[1] J. Jones, “Robots at the tipping point: The road to the iRobot roomba,” IEEE Robotics and Automation Magazine, vol.13, no.1, pp.76–78, 2006.

[2] H. Choset, “Coverage for robotics - a survey of recent results,” Annals of Mathematics and Artificial Intelligence, vol.31, no.1-4, pp.113–126, 2001.
[3] Y. Gabriely and E. Rimon, “Spanning-tree based coverage of continuous areas by a mobile robot,” Annals of Mathematics and Artificial Intelligence, vol.31, no.1-4, pp.77–98, 2001.

[4] N. Hazon and G.A. Kaminka, “Redundancy, efficiency and robustness in multi-robot coverage,” IEEE International Conference on Robotics and Automation, pp.735–741, 2005.

[5] N. Agmon, N. Hazon, and G.A. Kaminka, “Constructing spanning trees for efficient multi-robot coverage.” IEEE International Conference on Robotics and Automation, pp.1698–1703, 2006.

[6] X. Zheng, S. Koenig, D. Kempe, and S. Jain, “Multi-robot forest coverage for weighted and unweighted terrain,” IEEE Trans. Robotics, vol.26, no.6, pp.1018–1031, 2010.

[7] H.J. Quijano, “Improving cooperative robot exploration using an hexagonal world representation,” IEEE Fourth Congress of Electronics, Robotics and Automotive Mechanics, pp.450–455, 2007.