Cupp-E

The cup collecting robot

Participants	Kent Stark Olsen, B.Sc. student, Robot Systems Engineering The Maersk Mc-Kinney Moller Institute
	Lasse Simmelsgaard Børresen, B.Sc. student, Robot Systems Engineering The Maersk Mc-Kinney Moller Institute
	Leon Bonde Larsen, B.Sc. student, Robot Systems Engineering The Maersk Mc-Kinney Moller Institute
	Rudi Hansen, B.Sc. student, Robot Systems Engineering The Maersk Mc-Kinney Moller Institute

Deadline	2012-mm-dd

Faculty of Engineering
University of Southern Denmark
Niels Bohrs Allé 1
5230 Odense M
Denmark

www.sdu.dk/tek
+45 6550 7303
tek@tek.sdu.dk

Chapter 1

Summary

As data has been collected and empirically analysed by the canteen personal, found that researchers and students are lazy and will not bring their used service back to the canteen for cleaning. To accommodate this problem a robot is being designed which is able to collect "forgotten" cups and while performing this task also sweeps the floors. The robot platform has been developed but algorithms for the motion planning is still needed. This summary will give an overview of the software design for the robot.

Chapter 2

Coverage

2.1 Strategy

The coverage method used is boustrophedon, meaning “the way of the ox” from the plowing pattern of old days. The advantage of this strategy is that it ensures full coverage of a room with no obstacles and it is simple.

It is clear that when actually implemented on a robot, it must be supported by online sensors to cope with the obstacles present in the room, but for the purpose of estimating the distance travelled by such a robot, it gives a best case route.

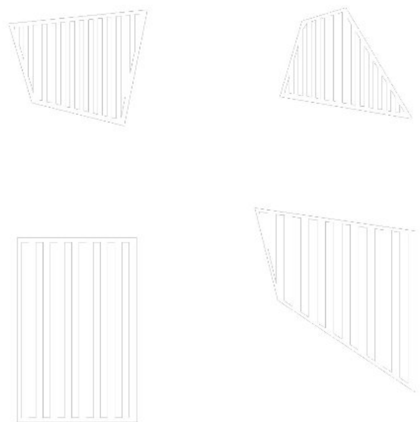

Figure 2.1: Shows example cells covered

2.1.1 Algorithm

The algorithm developed for covering follows the points listed in 1. The algorithm assumes the cell has been shrunk so the area given in the cell structure represents the free spae in which the robots center can move.

The algorithm first constructs a polygon from the cell data. Then the cell is swept by discrete vertical lines for every distance given by radius. This way a list is constructed holding the points where the sweeplines intersect the cell polygon, thus giving the robots turning points. Finally the list is sorted to reflect the order in which the turning points should be visited.

Algorithm 1 Boustrophedon coverage

```
Input:
cell                                   ▷ Structure holding vertices and edges
radius                                 ▷ The radius covered by the robot
list                                   ▷ empty list for returning the route
Output:
distance                               ▷ the lapsed distance
function COVER AREA(cell , radius , points)
    for each edge in cell do
        push line segment to polygon   ▷ This constructs a polygon shaped as the cell
    end for
    while sweepline inside polygon do
        construct next sweepline
        find intersections with polygon
        push intersections to points
    end while
    for each point in points do
        order points                   ▷ up,down,up,down sequence
    end for
    for each point in points do
        remove dublicates              ▷ in case intersection is a vertice
    end for
    for each point in points do
        calculate distance
    end for
    return distance
end function
```

2.1.2 Running time

The running time of this algoritm depends on the number of turningpoints in the returned path and theese again depends on the width of the cell and the radius of the robot.

$$n = 2 \cdot \frac{width}{radius} \tag{2.1}$$

Finding the point requires finding the possible intersections between a sweepline and the cell polygon. Since most cells are polygons of four vertices this means on average checking each sweepline against four lines or two checks per turning point. Checking the intersection between two lines means solving the equaled line equasions and is thus done in constant time giving this problem a complexity of O(n).

Ordering the list means iterating the list and reordering the points not fulfilling the criteria. Since swapping two entries of a vector is done in constant time, the operation is done in linear time.

Checking for dublicates means iterating the list and for each point checking the rest of the list for similar points. This operation is done i quadratic time.

Calculating the travelled distance is done by iterating through the list and adding up all the distances. Finding the distance between two points is done in constant time and thus the complexity of this problem is O(n).

2.1.3 Conclusion

Overall analysis of the algorithm thus shows that it runs in quadratic time due to the dublicate check and thus any effort to optimize the algorithm should start here, but since most cells are relatively small, the problem is also small.

CHAPTER 3

Online planning

3.1 Simulation

The simulation is done online, but seeks to reduce the output to a minimum. The main task is to visit all cells in the map and sweep them. This is simulated as an online statemachine changing between different brahaviours of the robot.

3.1.1 Behaviours

There are basically three beahaviours of the robot:

- Go to location
- Cover area
- Collect cup

The main task is solved using theese behaviours to visit all cells, covering the cells, collecting the cups when inside range and returning the cups when ever the tray is full.

3.1.2 State machine

The online simulation is done with a state machine facilitating the defined behaviours. The state machine follows the diagram in figure 3.1.

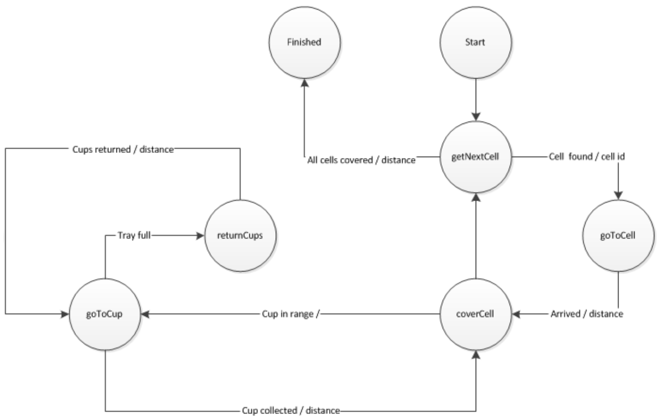

Figure 3.1: State machine diagram