



UNIVERSITY OF SOUTHERN DENMARK

## 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](http://www.sdu.dk/tek)  
+45 6550 7303  
[tek@tek.sdu.dk](mailto: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.

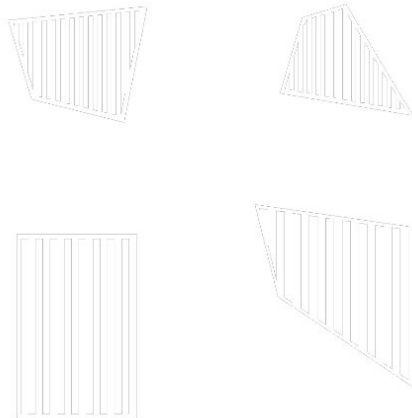


Figure 2.1: Shows example cells covered

#### 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 space in which the robot's 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 sweep lines intersect the cell polygon, thus giving the robot's 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

```

1: function COVER AREA(cell , radius , points)
2:   for each edge in cell do
3:     push line segment to polygon ▷ This constructs a polygon shaped as the cell
4:   end for
5:   while sweepline inside polygon do
6:     construct next sweepline
7:     find intersections with polygon
8:     push intersections to points
9:   end while
10:  for each point in points do
11:    order points ▷ up,down,up,down sequence
12:  end for
13:  for each point in points do
14:    remove duplicates ▷ in case intersection is a vertice
15:  end for
16:  for each point in points do
17:    calculate distance
18:  end for
19:  return distance
20: end function

```

---

### 2.1.2 Running time

The running time of this algorithm 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.

$$n = 2 \cdot \frac{width}{radius} \quad (2.1)$$

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 duplicates 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:

- Select a destination
- Go to a destination
- Cover a cell

The main task is solved using theese three behaviours encapsulated in a state machine. The state machine concists of two parts. The first part is a decision maker that selects a route and the other is an onlie stepper to follow the route.

#### 3.1.2 The decision maker

The decision maker selects a route based on the current state and is thus a state machine. In either state it facilitates the behaviours to select a route implemented as a list of points to visit followed by a state transition based on state variables. The state machine follows the diagram in figure 3.1.

#### 3.1.3 The stepper

The stepper is a sequential piece of code that iterates through the list of points adding up the travelled distance, checking for cups in range, collects the cups and returns the cups when tray is full. The task of returning the cups only contributes to the travelled distance since the robot cannot collect cups and the concept is therefore implemented as a mere calculation. A flow chart for the stepper can be seen in figure 3.2

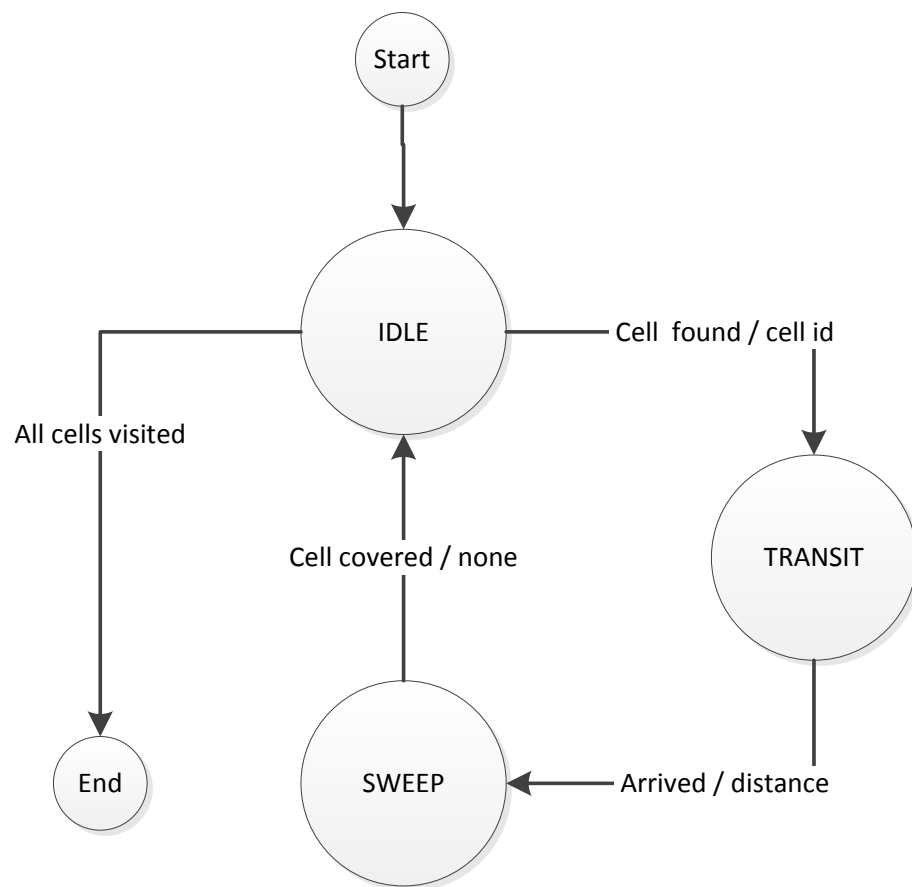


Figure 3.1: State machine diagram of the decision maker

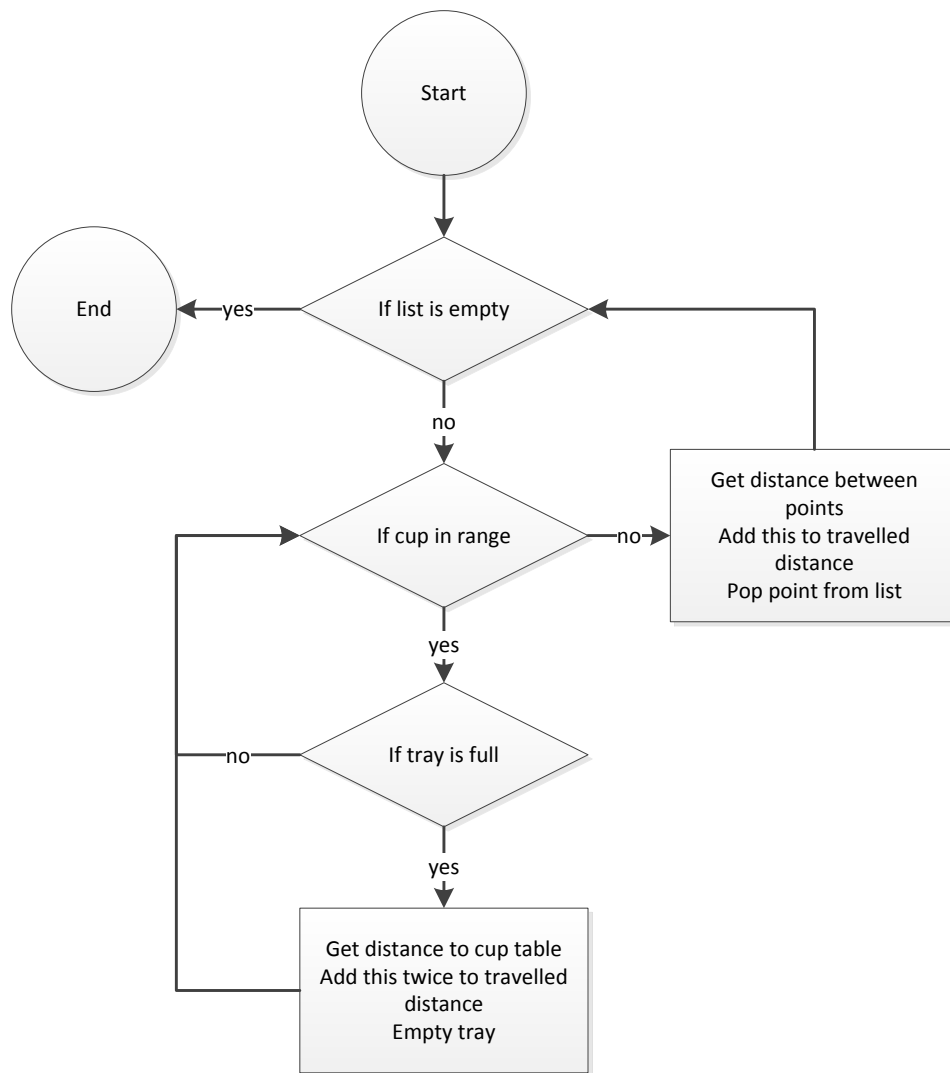


Figure 3.2: Flow chart of the stepper