From Coverage Path Planning to parking lot exploration

Anna Barbara Ádám, László Kocsány, Emese Gincsainé Szádeczky-Kardoss,
Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
Budapest, Hungary
Email: annadam97@gmail.com, kocsany.laszlo@gmail.com, szadeczky@iit.bme.hu

***Abstract*—Nowadays, it is getting really difficult to find an adequate free parking space, as there are more and more vehicles travelling on the roads. It is especially a difficult task to find a free parking place in huge parking garages and shopping malls, because the area of the parking zone is big, and there are a lot of cars searching for a free space. In some parking lots, a parking system is installed. These parking systems contain sensors at every parking space in order to perceive and indicate the occupancy of the given parking space. Using the signals of these systems, the vehicles can be navigated directly to the given parking space, for example using an IoT system. However, most of the parking lots are not equipped with these parking systems, so the driver must circumnavigate them, in order to be able to detect the free parking spaces. For an autonomous vehicle, a path should be planned, which drives along all the possible free parking spaces, so that the sensors of the vehicle could detect them. The parking lot exploration is very similar to the coverage path planning (CPP) problems. This paper presents CPP algorithm based parking lot exploration methods, in which personal preferences can be taken into account, too.**

***Index Terms*—Parking lot exploration, Autonomous vehicle, Coverage Path Planning, Cell decomposition**

I. Introduction

Nowadays, it is a time-consuming task to find a free parking space, especially in crowded shopping malls, parking garages, and in downtown. The main problem is that there is no efficient exploration strategy to find the most appropriate parking place in the shortest possible time.

To ease the finding of a free parking space and shorten the time drivers spend with it, different sensors are installed in modern parking garages. The installed sensors are mainly for detecting the presence of a car at the parking space. For example, magnetic sensors can detect the car body made of metal, pressure sensors installed in the floor can measure the weight of the vehicle, ultrasonic or infrared sensors can determine if something is in the examined area [1]. An IoT (Internet of Things) system can be built-up based on the signals of the sensors. The IoT system may also involve mobile applications, which indicate the free spaces to the driver [2]. The main disadvantage of these systems is the need for extra infrastructure, as these systems require sensors, power supply, possibly internet connection and maintenance.

The research was supported by the EFOP-3.6.2-16-2016-00014 and EFOP-3.6.2-16-2017-00002 projects - financed by the Ministry of Human Capacities of Hungary.

As most parking lots are not installed with these sensors, the chauffeur does not know where the free parking spaces are. Consequently, an explorational path should be planned around the parking lot in order to drive by all the possible free parking spaces. There can be preferred places in the parking lot, like the entrance of the shopping mall, the elevator, etc. It is manifest to look for parking spaces near the preferred locations at first and go further if there is no free parking space. The chauffeur decides whether the free parking space is acceptable or not based on the distance from the preferred places and the travelled route length.

The paper is organized as follows: Section II describes the most common Coverage Path Planning methods, some of which can be used during parking lot exploration.

Section III proposes the use of rectangular cell decomposition method, which is derived from trapezoidal cell decomposition. This method creates rectangular cells with smaller area, so it can be easily used for parking lot exploration.

Section IV presents the method, how wavefront algorithm can be used in order to plan the traversal of the created cells. There can be locations with different preferences in the map, and the preference values assigned to the cells can take this into account. The final cost function assigned to the cells determines if a parking space is adequate or not.

Section V derives conclusions from the presented methods, and presents the possible future work.

II. Coverage Path Planning Methods

The coverage path planning [3] is a group of path planning methods, in which a robot or vehicle visits all the free places in a given configuration. The purpose of these methods is to reach all the free points while avoiding the obstacles on the shortest path. Coverage path planning is used in many applications, such as robot vacuum cleaners, lawn mowers, painter robots and window cleaners. [3]–[5] This paper presents a method, which explores the parking lot by driving around all of the possible parking spaces. Parking spaces are considered as obstacles, so the algorithm only leads through the road surfaces around them.

A. Cell decomposition based path planning

There are several existing coverage path planning methods. Some of the methods divide the map into various sized cells.

The exact cellular decomposition [6] methods divide the free space into cells. The cells do not overlap, and the union of them covers the whole free space. One of the most commonly used exact cellular decomposition methods is trapezoidal cell decomposition. This method divides the map containing polygonal obstacles into trapezoidal or triangular cells. This method can only be used in case of polygonal obstacles, because the borders of the cells are created by extending rays from the vertices of the obstacles.

Boustrophedon (greek: ”ox turning”) decomposition [7] divides the map into larger cells, than trapezoidal cell decomposition, as this method only extends rays from the entry and exit points of the obstacles.

In Morse decomposition [6] critical points of Morse functions indicate the location of cell boundaries. A Morse function is a smooth function with only non-degenerate critical points.

The greedy convex polygon decomposition [8] divides the map into convex polygonal cells. There are two types of cuts: a single cut is a cut from a noncovex vertex to an existing edge or another cut, cutting two nonconvex vertices is called matching cut. First, the algorithm greedily searches for all the nonconvex vertices in order to make the matching cuts, then it makes single cuts for the unmatched nonconvex vertices. The partition edges come out from the set of the cuts and the boundary edges.

After creating the cells, the adjacency matrix of the cells can be created. Two cells are adjacent, if they have a given number of common points. It is possible to traverse from one cell to another if they are adjacent. The purpose is to visit all the cells in the map, but visit one cell only once. When the order of the cells is created, the traversal inside the cell should be planned.

B. Grid based path planning

Other coverage path planning methods are based on same-sized cells, called grid. The grid-based methods firstly divide the map into smaller, same-sized cells, then classifies the cells into two groups: the ones, which have no obstacles in them, and the others which contains any point of the obstacles.

The wavefront algorithm [3] needs a start and a goal cell. A distance value is assigned to each cell, starting from the goal point. The distance-value of the goal cell is 1, then every neighboring cell gets one bigger value. The cells are visited based on the distance values: the unvisited cell with the highest value is the next cell, if more unvisited cells have the same distance values, one of them is selected randomly.

The Spiral Spanning Tree Coverage (SpiralSTC) method [9] divides the map into same-sized cells. The obstacle free cells build up a graph, whose nodes are the centers of the cells, and the edges are the lines between the adjacent cell centers. A spanning tree is built up for this graph. This algorithm divides every cell into four identical subcells. The cell, which has four unvisited subcells, is a new cell, and the cell, which has at least one visited subcell is an old cell. At the beginning every subcell is unvisited. The algorithm starts at a given starting cell and marks this cell as an old cell, this cell is the root of the spanning tree. Then, every neighbor of this cell is tested in counterclockwise order if it is a new, obstacle free cell. While the current cell has new, obstacle free neighbors, the following steps are repeated: a spanning-tree edge is added from the current cell to the neighboring cell, then move to a subcell of the neighboring cell by following the right-side and add the center of the neighboring cell to the tree as a node. If there is a move-back from the current cell to a subcell of the parent cell along the right-side of the spanning tree edges, the algorithm terminates. This algorithm is performed recursively for each cell in the map.

C. Using coverage path planning methods for parking lot exploration

The basic ideas of coverage path planning methods can be used during parking lot exploration. The problem is similar because the purpose of parking lot exploration is to traverse every route, which drives by the possible free parking places, like in coverage path planning the planned path goes through all the free points of the workspace. The main difference is the following: during the parking lot exploration, it is not necessary to visit every obstacle free point. Planning the traversal of the cells is enough for the exploration, the traversal inside each cell is not needed to be planned.

The main idea is to divide the map into smaller cells, then plan a traversal, which goes through all the cell and minimizes reversals. The solution that seems the best, is to use the trapezoidal cell decomposition, as most of the maps consist of polygonal obstacles, so this method is applicable. For planning the traversal of the cells, the wavefront algorithm can be used with a little modification: the cells are treated as grid and the start and goal cells are the same. The advantage of this method is that, the distance values can be modified in order to take personal preferences into consideration. The details of this method can be found in the following sections.

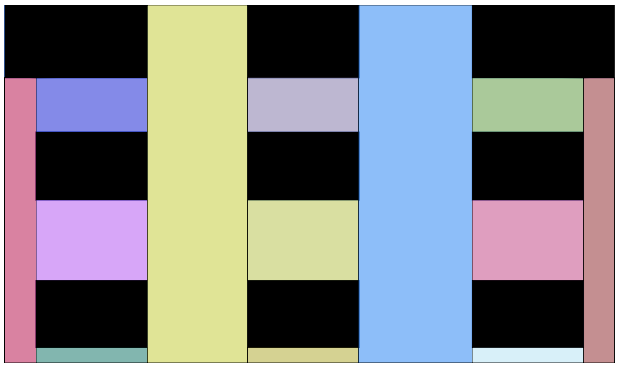
III. Rectangular cell decomposition

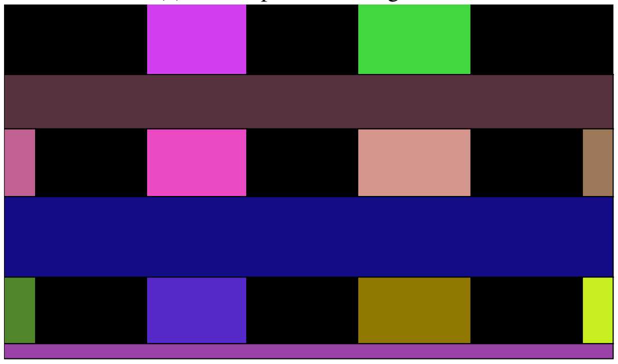
Trapezoidal cell decomposition can be used in polygonal environments, but it is not a strict requirement, as most of the parking lots consist of polygonal obstacles. If there are non-polygonal obstacles in the map, trapezoidal cell decomposition can be applied after preprocessing, e.g. using the bounding boxes of the obstacles. By applying the trapezoidal cell decomposition, the map is only decomposed by x or y axis. It is more advantageous to create cells with smaller areas, so it is manifest to decompose the map by x and y axes, too. The intersections of the cells decomposed by x and y axes make up a new decomposition (see Fig. 1a-1c). This decomposition leads to rectangular cells with smaller areas. [10]

Another advantage of this method is that every cell has only one neighboring cell in each direction, so a traversal can be planned which avoids reversing when it is possible.

IV. Wavefront algorithm based traversal of the cells

Wavefront algorithm can be used in order to determine the traversal of the cells [10], [11]. Wavefront algorithm is applied


(a) Decomposition along x axis


(b) Decomposition along y axis

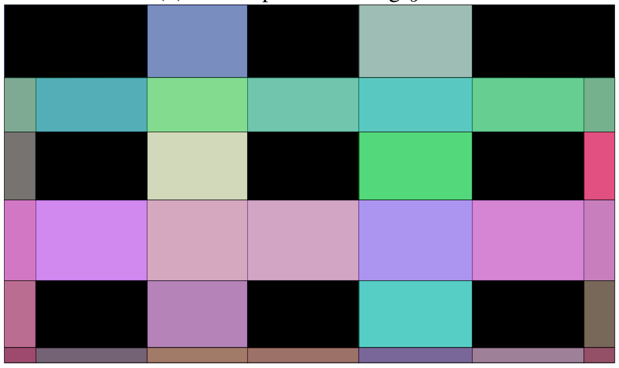

(c) Intersections of the cells from decompositions along x and y axis

Fig. 1: Rectangular cell decomposition (black areas represent the obstacles and the colorful areas represent the road surfaces)

in grid-based Coverage Path Planning. Grid is originally built up from same-sized cells, so in this approach the decomposed map can be treated as a grid, the only difference is the fact, that the cells are of different size. The algorithm assigns distance values to the cells: it assigns 0 to the start cell, and assigns one bigger value to the neighboring cells recursively until there are unmarked cells left. An example of the distance values can be seen in Fig. 2.

Preference values can be added to distance values (see Section IV-A) and various exploration strategies can be used (see Section IV-B-IV-D).

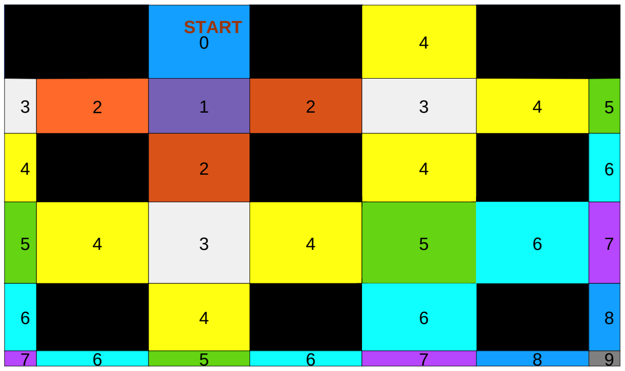


Fig. 2: Distance values assigned to the cells

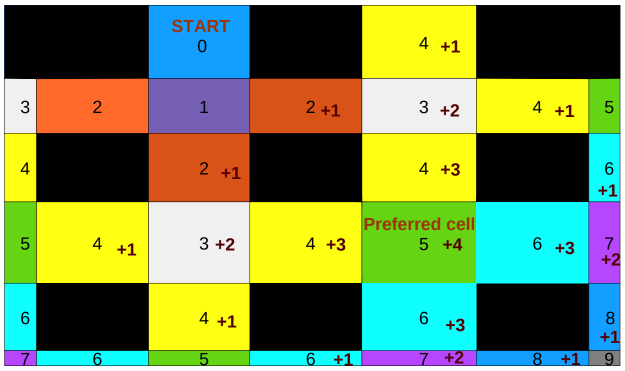


Fig. 3: Distance values and additional preference values when applying additional preference (value of 4 and range of 4), normal numbers represent the distance values, bold numbers represent the additional preference values

A. *Preference assigned to cells*

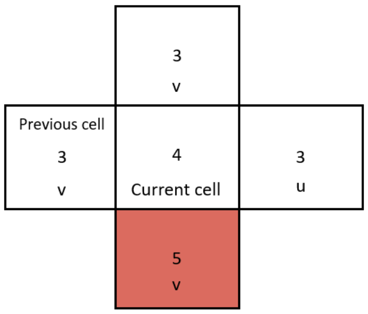
Some locations are preferred over others, therefore the above-mentioned distance values can be modified so that the more preferable locations are visited first and more frequently. A preference value is added to the distance value of the cells which are near to the preferred location. The preference value is based on personal preferences (e.g. entrances, elevators in shopping malls). The added preference value attracts the path from a given range.

The cell, which is the nearest to the preferred location gets the maximum of the preference value, and every neighboring cell gets one smaller preference value. Cells, which are outside the given range do not get additional preference value (see Fig. 3).

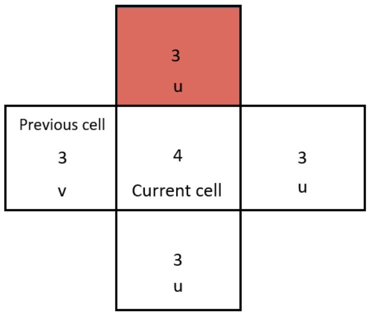
It is possible, that more than one cell is preferred. In this case, the additional preference values can overlap, so a cell, which is in the range of more preferred cells get the sum of the preference values. Furthermore, it is possible that some locations are to avoid, so negative preference values are also permitted, which toss the path away from them.

B. *Preference based traversal*

The traversal starts from the starting point and the wavefront algorithm firstly visits those neighboring cells, which have the



(a) Preference value determines the following cell



(b) The testing order of the cells determines the following cell

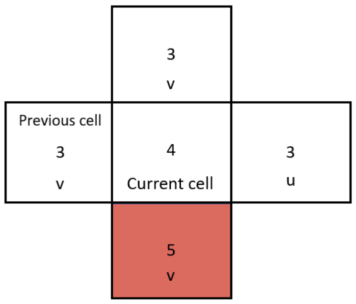
Fig. 4: The numbers represent the preference values of the cells, there are visited (v) and unvisited (u) neighboring cells, the following cell is colored red when applying preference based traversal

highest preference value.

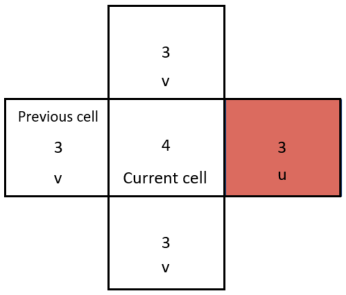
This algorithm also tries to avoid reversing, so the reverse direction is the least preferred direction. If a cell has adjacent cells of the same preference value, they are visited in the following order: left, right, upper, lower neighboring cell, so a lower priority is based on the directions (see Fig. 4). This method leads to a traversal, in which some of the cells may remain unvisited, but cells with high preference value become visited, and the planned path circle around these cells, until a free parking space is found.

C. Preference and visitedness based traversal

This algorithm mainly differs from the one introduced in Subsection IV-B in the management of adjacent cells with the same preference value. This algorithm also visits those neighboring cells first, which have the highest preference value and tries to avoid reversing. Though, if a cell has adjacent cells of the same preference value, the unvisited one will be the following cell. The visitedness is tested in the following order: left, right, upper, lower neighboring cell. The traversal visits the highly preferred cells more frequently, but in this case the lower priority is based on the visitedness of the cells.



(a) Preference value determines the following cell



(b) The visitedness of the cells determines the following cell

Fig. 5: The numbers represent the preference values of the cells, there are visited (v) and unvisited (u) neighboring cells, the following cell is colored red when applying preference and visitedness based traversal

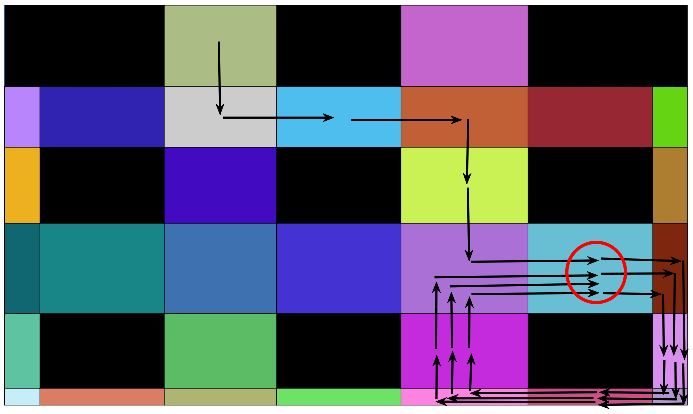


Fig. 6: Example of adjacency graph traversal in a map , there was added preference value in the cell, marked by a red circle, the order of the cells is based on preference and visitedness

So the unvisited one of the neighbors with the same preference value will be visited (see Fig. 5). Fig. 6 shows an example of the traversal.

D. Visitedness and preference based traversal

The algorithm firstly visits the unvisited neighbors of a cell. If a cell has more than one unvisited neighboring cell, the next

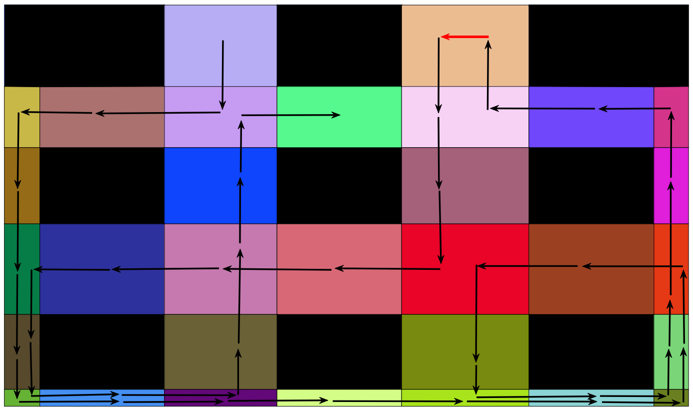


Fig. 7: Example of adjacency graph traversal in a map, red arrows represent the backtracks, there were no added preference values during the traversal, the order of the cells is based on visitedness and preference

cell is the one with highest preference value.(See fig. 8.) This algorithm plans a traversal, which firstly traverses the unvisited neighbors of the cells. Cells with high preference values are visited more frequently, but cells remain unvisited less often. This algorithm avoids reversals, when it is possible, too. Fig. 7 shows an example of a traversal.

E. Make the traversal repeatable

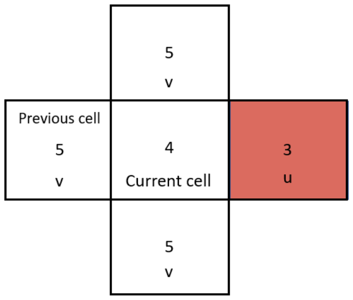
The traversals generated by the presented methods do not lead back to the initial cell. [10] Though, it often occurs, that there are no free parking spaces in a given parking lot, so in this case, the traversal should be repeated, until an adequate free parking space is found. In order that the traversal could be repeated, a path from the final cell to the initial cell must be planned. It can be realized by assigning a high preference value to the initial cell with a large range, in order to attract the path there from any cell. Another possible solution is to treat the final cell as the initial cell and regenerate the traversal this way.

F. Modifying the preference values during traversal

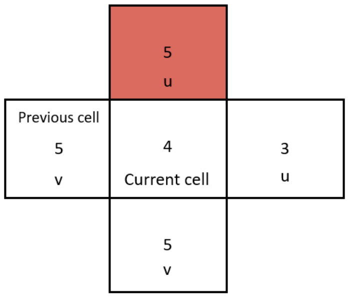
It is possible to modify the preference values during the exploration. It is a manifest solution to reduce the preference value of the cell by 1, when a cell is visited. This leads to a traversal, in which the visited cells attract the route less, so there is a bigger chance that the unvisited cells become visited. This traversal revisits the visited cells less frequently, so this solution is not applicable in cases, when the driver would like to park in a given position at all events.

G. Cost of the traversal

The goal of the parking space searching is to find a free parking space near to the preferred areas as soon as possible. Usually, in parking lots, it is very difficult to find a free parking space near to the appointed positions, so if the driver insists on the location, it is possible, that multiple rounds are needed around the parking lot. So there is a trade-off between driven route length and the nearness of the parking space.



(a) Visitedness of the cells determines the following cell



(b) The preference value of the cells determines the following cell

Fig. 8: The numbers represent the preference values of the cells, there are visited (v) and unvisited (u) neighboring cells, the following cell is colored red when applying visitedness and preference based traversal

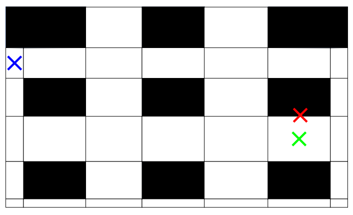
The route length ($RouteLength$) can be estimated by summing the distances between the centers of the following cells ($Length(Route_i)$). The nearness ($Nearness$) of the parking space is the preference value ($prefVal_i$) weighted sum of the distances of the preferred locations ($dist(position, prefLoc_i)$). These cost functions are calculated for each cell during the traversal. The final cost function ($cost$) is the weighted sum of the previously mentioned normalized cost functions. The weighting depends on the drivers preferences: the importance of nearness or shortness of the driven route.

$$RouteLength = \sum_{i=1}^{N} Length(Route_i) \tag{1}$$

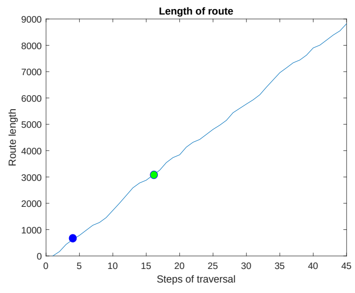
$$Nearness = \sum_{i=1}^{P} \frac{prefVal_i}{\sum_{i=1}^{P} prefVal_i} \cdot dist(position, prefLoc_i) \tag{2}$$

$$cost = \alpha \cdot RouteLength + (1-\alpha) \cdot Nearness \tag{3}$$

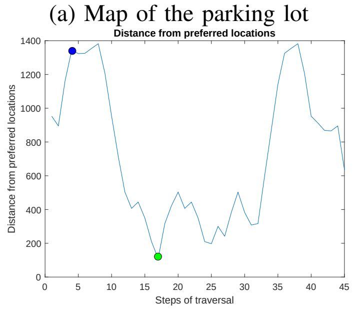
where N is the number of route sections, P is the number of preferred locations and α is weighting factor.



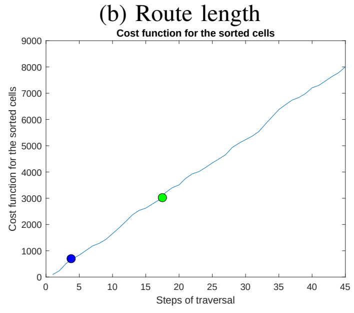
(a) Map of the parking lot



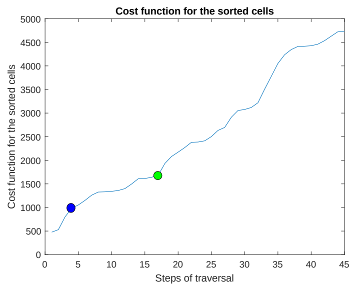
(b) Route length



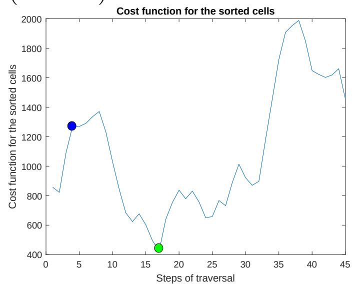
(c) Distances from the preferred location



(d) Weighted cost function ($\alpha = 0.9$)



(e) Weighted cost function ($\alpha = 0.5$)



(f) Weighted cost function ($\alpha = 0.1$)

Fig. 9: The subfigures show an example of the cost function with different α values

If a free parking space is found along the cell, the value of the final cost function determines whether it is an adequate parking space or not. A threshold value is needed to be established by the driver, parking spaces of a lower cost value are suitable. Fig. 9 shows an example of the cost function of the traversal shown by Fig. 7. In Fig. 9a the preferred location is marked with a red X, the example cells are marked with blue and green X. Fig. 9b shows the driven route length during the traversal, the values in the example cells are marked with colorful dots. It can be seen that the most preferable parking space depends on the α parameter. The smaller the α value is, the better a nearer parking space is.

V. Conclusion and Future Work

This paper presented more Coverage Path Planning (CPP) methods, some of which can be used for parking lot exploration. The main idea of CPP and parking lot exploration is the same: an area is given, which is needed to be explored. In CPP every point of the free workspace is needed to be visited, in contrast, during parking lot exploration, only an exploring path is needed, which drives by all the possible free parking places, so there is no need to visit every free point of the road surface.

An advanced version of trapezoidal decomposition can be used to divide the map of a parking lot into smaller, rectangular cells. The traversal of the cells can be planned using the base idea of wavefront algorithm: a distance value is assigned to every cell, and neighboring cells with the highest values are visited first, until all the cells are visited, or a terminal condition is met.

Since there are more and less preferred parking spaces in a parking lot, the distance values of the cells should be modified taking into consideration the preference of their location. By modifying the distance values, different traversals can be planned based on only preference, or preference and visitedness of the cells.

The algorithm decides whether a parking space is adequate or not based on a cost function. The cost function is built up from two parts: the driven route length and the distance between the given parking space and the preferred locations. These parts can be taken into account with different weighting, which gives the final cost of a parking space. If the cost of the current free parking space is below a given threshold, it is an adequate parking space and the vehicle can park there.

The algorithms were tested in Matlab environment, future work includes testing in real environment and developing the algorithm to be able to handle multi-storey car parks.

References

[1] Faheem, S.A. Mahmud, G.M. Khan, M. Rahman, and H. Zafar. A survey of intelligent car parking system. *Journal of Applied Research and Technology*, 11(5):714 – 726, 2013.

[2] Fadi Al-Turjman and Arman Malekloo. Smart parking in IoT-enabled cities: A survey. *Sustainable Cities and Society*, 49:101608, 2019.

[3] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258 – 1276, 2013.

[4] Sanghoon Baek, Tae-Kyeong Lee, Se-Young Oh, and Kwangro Ju. Integrated on-line localization, mapping and coverage algorithm of unknown environments for robotic vacuum cleaners based on minimal sensing. *Advanced Robotics*, 25:1651–1673, 01 2011.

[5] Ibrahim Hameed. Coverage path planning software for autonomous robotic lawn mower using dubins' curve. 07 2017.

[6] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz. Exact cellular decompositions in terms of critical points of morse functions. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2270–2277 vol.3, 2000.

[7] Howie Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Auton. Robots*, 9:247–253, 12 2000.

[8] Arun Das, Michael Diu, Neil Mathew, Christian Scharfenberger, James Servos, Andy Wong, John Zelek, David Clausi, and Steven Waslander. Mapping, planning, and sample detection strategies for autonomous exploration. *Journal of Field Robotics*, 31, 01 2014.

[9] Y. Gabriely and E. Rimon. Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 954–960 vol.1, 2002.

[10] A. B. Ádám et al. Cell decomposition based paring lot exploration. *Proceedings of the Workshop on the Advances of Information Technology*, pages 5–12, 2020.

[11] E. Galceran and M. Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.