**ACTA IMEKO**
**ISSN: 2221-870X**
*September 2021, Volume 10, Number 3, 15 - 27*

# Using coverage path planning methods for car park exploration

**Anna Barbara Ádám[1], László Kocsány[1], Emese Gincsainé Szádeczky-Kardoss[1]**

[1] *Department of Control Engineering and Information Technology, Budapest University of Technology and Economics, Budapest, Hungary*

ABSTRACT
With the increasing number of vehicles on the roads, finding a free parking space has become a time-consuming problem. Traditional car parks are not equipped with occupancy sensors, so planning a systematic traversal of a car park can ease and shorten the search. Since car park exploration is similar to coverage path planning (CPP) problems, the core concepts of CPP algorithms can be used. This paper presents a method that divides maps into smaller cells using trapezoidal cell decomposition and then plans the traversal using wavefront algorithm core concepts. This method can be used for multi-storey car parks by planning the traversal of each floor separately and then the path from one floor to the next. Several alternative explorational paths can be generated by taking different personal preferences into account, such as the length of the driven route and the proximity to preferred locations. The planned traversals are compared by step number, the cell visitedness ratio, the number of visits to each cell and the cost function. The comparison of the methods is based on simulation results.

**Section:** RESEARCH PAPER

**Keywords:** Car park exploration; coverage path planning; parking assistant system

**Citation:** Anna Barbara Ádám, László Kocsány, Emese Gincsainé Szádeczky-Kardoss, Using coverage path planning methods for car park exploration, Acta IMEKO, vol. 10, no. 3, article 5, September 2021, identifier: IMEKO-ACTA-10 (2021)-03-05

**Section Editor:** Bálint Kiss, Budapest University of Technology and Economics, Hungary

**Received** January 13, 2021; **In final form** September 17, 2021; **Published** September 2021



**Corresponding author:** Anna Barbara Ádám, e-mail: annadam97@gmail.com

## 1. INTRODUCTION

With an increasing number of vehicles on the roads, it is becoming difficult to find a free parking space. Several sensor-based parking assistant systems have been developed in the past decade to make it easier to find a free parking space in busy areas, such as city centres and shopping malls. These systems mainly include sensors installed in each parking space that can detect the presence of a car by measuring its weight with pressure sensors; sensing the car body with magnetic sensors or infrared and ultrasonic sensors can determine if something is in the examined area.

The main problem with these parking systems is the need for extra infrastructure and sensors. As most car parks are not equipped with sensors, which indicate the occupancy of the parking space, a vehicle must drive passed a parking space to be able to detect if it is free [1]. There are also Internet-of-Things (IoT) systems, which involve not only the signals of the sensors but also mobile applications [2]. These systems can navigate the driver to the free parking spaces in the shortest possible time.

The main purpose of this paper is to present an installed sensor-free car park exploration method that navigates the vehicle to all the possible free parking spaces. Autonomous vehicles are able to detect the free parking spaces with sensors installed in the vehicle (e.g. LiDAR [3]). As the coverage path planning (CPP) problem is similar to the car park exploration problem, the core concepts of CPP algorithms can be used.

This paper is organised as follows. Section 2 presents the most commonly used parking systems and provides examples, while the formulation of the car park exploration problem can be found in Section 3. Section 4 presents some cell decomposition and grid-based CPP methods and explains how they are used for car park exploration. An improved version of trapezoidal cell decomposition can also be found in this section. Different traversal methods are presented in Section 5, while Section 6 introduces a cost function to grade the free parking spaces. The presented exploration methods are compared in Section 7, and the method for using them in multi-storey car parks is presented in Section 8. Section 9 derives the conclusions from the different traversals and presents possible avenues for future work.

## 2. PARKING ASSISTANT SYSTEMS

The literature proposes various solutions for autonomous parking with sensors installed in car parks. These systems require a centralised parking system in which a server stores the occupancy of the parking spaces based on the sensor signals. When a vehicle requests a parking space, the server reserves one for the vehicle.

CCTV systems are widely available in car parks. Consequently, algorithms using image processing algorithms are able to detect the occupancy of a parking space based on camera signals. For example, Athira et al. [4] present an optical character recognition system that detects occupied parking spaces based on CCTV signals.

Another solution is the availability of IoT-enabled cities, often referred to as smart cities. Smart cities can provide information about the availability of parking spaces. If a centralised parking system server is available, the vehicle is able to request information about the occupancy of parking spaces from the server. Al-Turjman et al. [4] present a survey of IoT-enabled cities. The use-case practices are also presented, including the smart payment system, the parking reservation system and the e-parking system.

There are several commercially available solutions for smart parking. In Hungary, two such solutions can be found in Budapest. The first is Parkl [5], which is a smartphone application providing information about the location of car parks and making cashless payment possible. Parkl does not provide the exact location of the parking spaces but the location of parking zones in the city where possible parking spaces can be found. In comparison, Parker [6] (developed by Smart Lynx Ltd.) provides information about the exact occupancy of about 1,500 parking spaces in the city centre. This solution uses preinstalled sensors to indicate the occupancy of a given parking space to the driver. Parker also provides a cashless payment service for its users.

The algorithm presented in this paper provides a car park exploration method using CPP. Car park exploration is required because no additional signals for preinstalled sensors in car parks are used. A complete system is therefore required to perform the car park search from the creation of the exploration path to detecting appropriate parking spaces and performing the parking manoeuvre. This system is called the autonomous parking system and consists of four subsystems, as the parking task can be broken down into four subtasks.

The first task of the autonomous parking system is to provide an exploration path for the vehicle. The focus of this paper is to propose a solution for this subsystem by applying CPP algorithms. This subsystem provides multiple goal configurations for the vehicle. These goal configurations are required, as the avoidance of preinstalled sensors leads to the loss of the goal configuration.

The second subsystem is a detector subsystem. The main task is to scan the environment and detect parking spaces for the vehicle while the vehicle is driving along the exploration path [3]. In order to be able to detect parking spaces, a sensor must be mounted on the vehicle. The first subsystem should consider the sensor parameters during the planning of the exploration path.

When a parking space is found, the third and fourth subsystems plan and execute the parking manoeuvre [7], [8].

## 3. CAR PARK EXPLORATION

The goal of car park exploration is to plan a path leading to all the possible free parking spaces. The binary map (Figure 2) of the car park (Figure 1) is known, and the parking spaces are treated as obstacles. While following the planned path, a sensor (e.g. LiDAR) searches for a suitable free parking space.

Figure 1. Map of a car park.

Figure 2. Binary map of a car park.

In this formulation, $\mathrm{C} \subset \mathrm{R}^2$ defines the workspace of car park exploration and $\mathcal{A}$ denotes the vehicle. The state of the vehicle is $q = \begin{bmatrix} x \\ y \end{bmatrix}$, where $\mathcal{A}(q) \subset C$, while $\begin{bmatrix} x \\ y \end{bmatrix}$ denotes the position of the vehicle in a fixed frame (the orientation of the vehicle is not taken into account). The workspace consists of obstacles ($C_{\mathrm{obs}} \subset C$) and free spaces ($C_{\mathrm{free}} = \mathrm{C} \setminus C_{\mathrm{obs}}$), some of which need to be visited ($C_{\mathrm{vis}} \subseteq C_{\mathrm{free}}$). The vehicle can move only in free space ($\forall q \in C_{\mathrm{free}}$).

The vehicle moves on a collision-free path ($\tau$), where $s_i \in R$ is a scalar path parameter ($s \in [0, T]$, $T$ is the length of the whole path):

$$\tau: s \mapsto q, \qquad \forall s \in [0,T]: \tau(s) \in C_{\mathrm{free}}\,. \tag{1}$$

$L(q) \subset C$ denotes the points that are inside the range ($\delta$) of the sensor, which detects the free parking spaces:

$$L(q) = \{\, z \in C_{\mathrm{free}} |\ \|q - z\| \leq \delta\}, \tag{2}$$

where $\|q - z\|$ is the Euclidean distance between points $q$ and $z$.

The points seen while traversing the path are

$$L(\tau(t)) = \bigcup_{s \in [0,t]} L(\tau(s)). \tag{3}$$

The aim is to reach every position that should be visited during the exploration:

$$C_{\mathrm{vis}} \subseteq L(\tau(T)). \tag{4}$$

Other constraints that should be considered are as follows:

- The start position is $\tau(0) = q_{\mathrm{init}}$.

- Cost can be assigned to the path as $w_1(\tau)$ (e.g. length of the path).
- Preferred positions are provided for, which are considered first. The $w_2(q)$ cost can be assigned to the $q \in C_{free}$ position (e.g. distance from the target position).
- The traversal can be interrupted when a condition is met (parking space detected with LiDAR).
- When stopping while driving (at some point with $s_1$ path parameter), the cost of the traversal is the personal preference weighted ($\alpha$) sum of $w_1$ and $w_2$: $w_1(\tau) + \alpha\, w_2(\tau(s_1))$ (the cost of the path up to that point plus the cost of the position).
- There might be constraints to the order of the configurations in $\tau$ path (e.g one-way streets).

## 4. USING COVERAGE PATH PLANNING METHODS

The purpose of CPP methods is to plan an obstacle-free path that reaches every free point of the given configuration.

This section presents car park traversal-related computational problems. It also explains the idea behind using CPP algorithms for the traversal and provides an overview of the most commonly used cell decomposition and grid-based methods. Finally, methods chosen for implementation are presented.

### 4.1. Related computational problems

CPP is similar to the travelling salesman problem [10], in which the salesman has to visit all the cities via the shortest route possible and return to the beginning. The cities are represented as nodes on a graph and the routes between them are the edges. The travelling salesman problem is NP-hard, which means it is at least as hard as any NP problem, so it cannot be solved with polynomial-time algorithms.

During path planning, an important factor in terms of the path is the exploration of the environment. Two computational geometric problems are related to this: the museum problem [11] calculates the fewest number of guards required to observe the whole museum, while in the watchman problem [12], only the map of the region is given, and the main purpose is to plan the shortest possible path between the obstacles so that the watchman can guard the entire area.

### 4.2. Cell decomposition-based path planning

There are several cell decomposition-based path planning methods that divide a map into cells. Exact cellular decomposition methods [12] divide the free space into various-sized nonoverlapping cells. The union of the cells covers the whole free configuration.

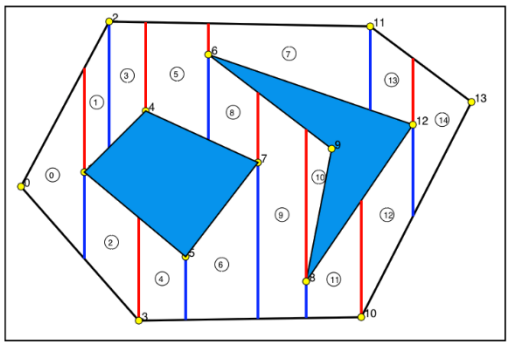

Figure 3. Example of trapezoidal cell decomposition [14].

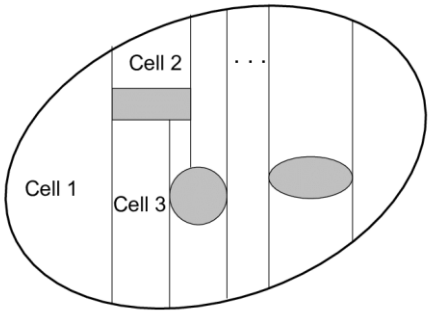


Figure 4. Example of Boustrophedon decomposition [16].

Trapezoidal cell decomposition [14] can only be used in polygonal environments, as it extends rays from the vertices of the obstacles. These rays and the edges of the obstacles form the cell borders, dividing a map into trapezoidal or triangular cells. An example of this method can be seen in Figure 3.

Boustrophedon (Greek: 'ox turning') decomposition [15] extends rays from the entry and exit points of the obstacles, so fewer rays are extended than in trapezoidal cell decomposition. Consequently, the cells have a larger area. Figure 4 shows an example of this method.

When applying Morse decomposition [12], the critical points of the smooth function, called the Morse function, indicate the boundaries of the cells (see Figure 5).

Greedy convex polygon decomposition [18] can be used when there are polygonal obstacles. This method consists of two types of cuts:

- a single cut: a cut from a nonconvex vertex to an existing edge or another cut,
- a matching cut: cutting two nonconvex vertices.

First, all the matching cuts are made on the nonconvex vertices, then the single cuts are made for the unmatched vertices. In the example in Figure 6, the matching cuts are green and the single cuts are red. The cell boundaries are the set of cuts and the edges of the obstacles.

After dividing the map, the adjacency matrix of the cells can be created. Two cells are adjacent if they have a common boundary, and the boundary has a given number of common points. The purpose of cell decomposition-based methods is to visit every cell once, although it is not always possible. If the adjacency matrix of the cells is known, a traversal can be planned, as it is known which cell can be visited from the current one. After creating the traversal, a path can be planned that leads through the cells in a given order reaching every free point of the configuration.

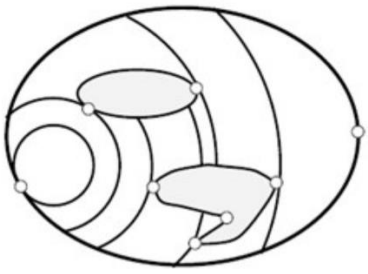

Figure 5. Example of Morse decomposition [17].

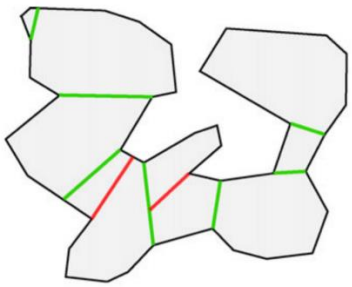

Figure 6. Example of greedy convex decomposition [18].

### 4.3. Grid-based path planning

In contrast to the cell decomposition-based methods, grid-based methods divide the map into same-sized cells, called a grid. The cells are classified into two groups, those that have obstacle points inside the cell and those without obstacle points.

The wavefront algorithm [9] assigns distance values to every cell in the map, with each neighbouring cell being assigned one larger distance value, starting from the initial cell, which has a distance value of 0. The traversal of the cells is based on distance values, and the neighbouring unvisited cell with the largest distance value is the following cell. If a number of unvisited neighbouring cells have the same distance value, the following cell is selected randomly. An example of this method can be seen in Figure 7.

The spiral spanning-tree coverage method [21] builds up a graph with nodes that represent the centres of the obstacle-free cells and edges that represent the lines between the neighbouring cell centres. The cells are divided into four subcells, which are classified into two groups: those that have four unvisited subcells (called new cells) and those that have at least one visited subcell (called old cells). First, every cell is unvisited, and the algorithm starts from the initial cell and marks it as an old cell. The initial cell is the root of the spanning tree. In the subsequent steps, every cell neighbouring the current cell is tested if it is a new cell. If the cell has a neighbouring new cell, a spanning-tree edge is added from the current cell to the neighbouring cell, and then the algorithm moves to a subcell of the neighbouring cell and adds the centre of the neighbouring cell to the tree as a node. If there is a backwards move from the current cell to a subcell of the parent cell along the right-hand edge of the spanning tree, the algorithm terminates. Figure 8 provides an example of this method.

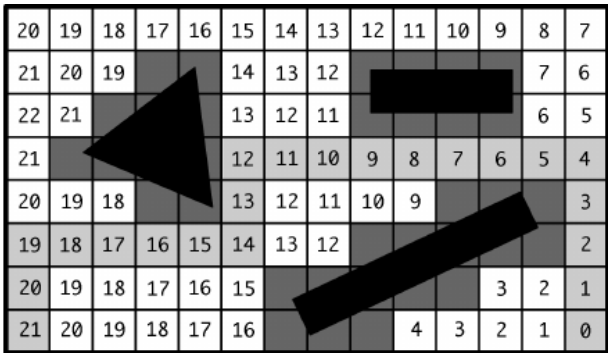


Figure 7. Example of a wavefront algorithm [19].

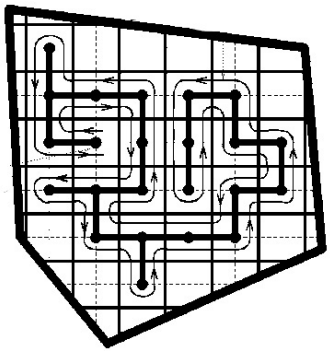

Figure 8. Example of spiral spanning-tree coverage [22].

### 4.4. Using coverage path planning methods for car park exploration

In order to plan the traversal of a car park, the map of the car park should be divided into smaller regions. As most car parks consist of polygonal obstacles, trapezoidal cell decomposition can be used to divide the map. The cells can be treated as a grid, so wavefront algorithm core concepts can be used to plan the traversal of the cells. The main advantage of the wavefront algorithm is that it can take personal preferences into account by modifying the distance values of the cells.

### 4.5. Rectangular cell decomposition

As rectangular decomposition is based on trapezoidal cell decomposition, it can only be used in polygonal environments. If the map contains nonpolygonal obstacles, the bounding boxes of the obstacles should be taken into account when decomposing the map. Trapezoidal cell decomposition decomposes the map using only one axis ($x$ or $y$), so the decomposed map may contain cells covering large areas. The main disadvantage of this method is that the traversal of the cells is not unequivocal, so a path should be planned inside a cell from which all of the free parking spaces can be seen.

Rectangular cell decomposition decomposes the map using both $x$ and $y$ axes. The final cells are the intersections of the cells decomposed by these axes. The final cells are smaller and rectangular, which is the main advantage of this method; every cell has only one neighbouring cell in each direction. An example of a decomposed map can be seen in Figure 9, where the coloured rectangles indicate the cells.

The adjacency matrix of the cells can be created after dividing the map. This matrix is an $n \times 4$ matrix, where $n$ is the number of cells. The rows of the matrix store the indices of the adjacent cells of every cell in each direction. If two cells have common boundaries, and the vehicle can pass from one cell to the other, these cells are adjacent. Cells neighbouring obstacles and cells on the edges of the map do not have four neighbouring cells, so the neighbouring obstacles and edges are considered as their neighbours (an implementational solution might be to store these false neighbours as dummy indices, e.g. −1). If there are one-way road sections, moving from one cell to the other is permitted, but moving in the opposite direction is prohibited, so the $i^{th}$ cell is adjacent to the $j^{th}$ cell but not vice versa. This means that the $i^{th}$ row of the adjacency matrix contains the $j$-cell index in the appropriate column, but the $j^{th}$ row contains a dummy index instead of the $i$ index.

More details of this algorithm can be found in [20].

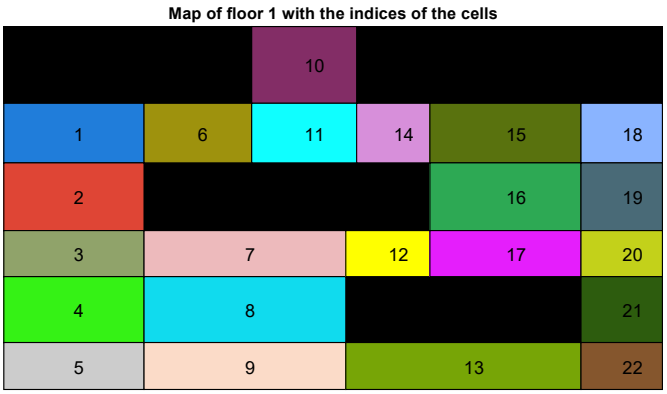


Figure 9. The decomposed map of the car park; the initial cell is cell number 10.

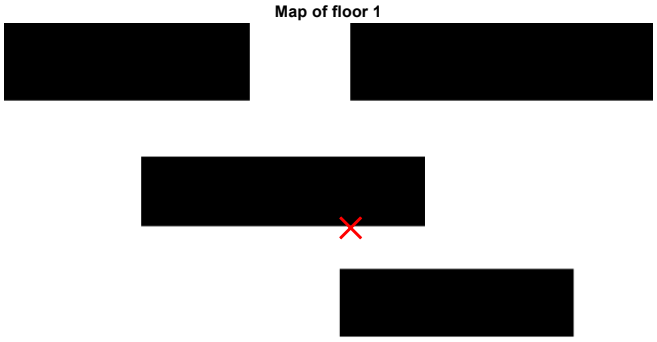


Figure 10. Map of the car park; the red X represents the preferred location.

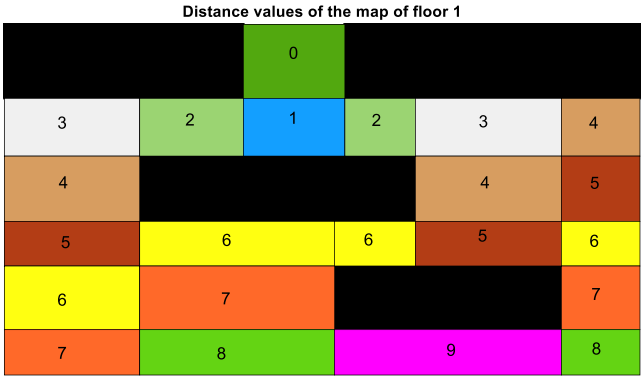


Figure 11. The distance values of the cells.

## 5. TRAVERSAL OF THE CELLS

Several different traversal methods are presented in this section. The map presented in Section 5.1 is used as an illustration.

### 5.1. Map of the car park

The car park consists of only one floor, with the black areas representing the obstacles (the possible free parking spaces are also considered as obstacles). The red X represents the preferred location. The map of this car park can be seen in Figure 10, the decomposed map can be seen in Figure 9 (rectangular decomposition was used) and the assigned distance values from the wavefront algorithm can be seen in Figure 11.

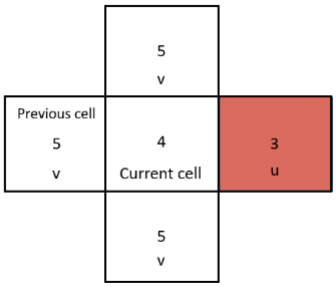


Figure 12. The visitedness of the cells determines the following cell, which is in red.

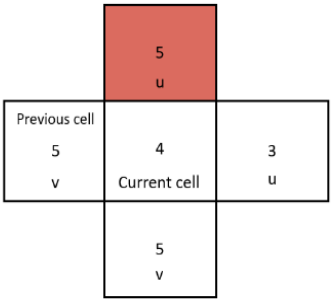


Figure 13. The preference value determines the following cell.

All the algorithms stop when the step number exceeds a given number, which depends on the number of cells. The maximal step number of the following simulations is 57.

### 5.2. Visitedness- and preference-based traversal

This method first visits the unvisited neighbouring cells (see Figure 12). If there are a number of neighbouring cells that are unvisited, the cell with the highest preference value is the following cell (see Figure 13).

In the map in Section 5.1, one cell remains unvisited. This results from the low preference values in that area. The algorithm visits the neighbouring cells of the unvisited cell only once, and the other cells have higher preference values, so they are the next cells of the traversal. A traversal designed by this method can be seen in Figure 14.

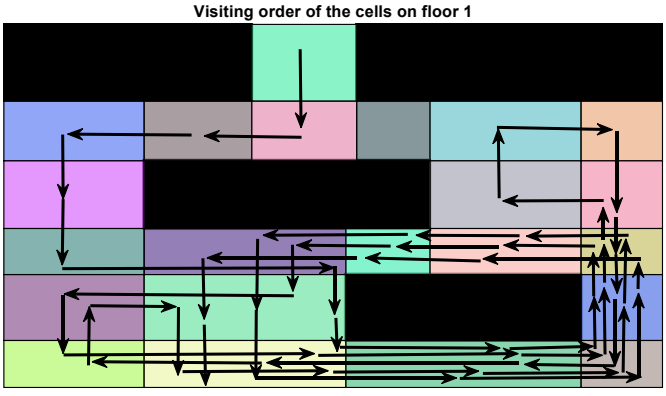


Figure 14. The visitedness- and preference-based traversal of the car park.

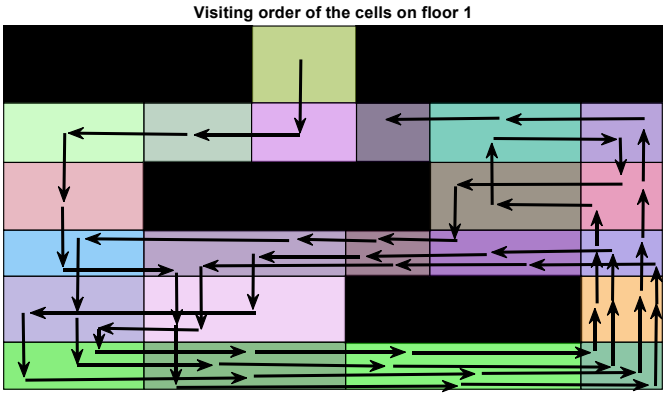


Figure 15. The visitedness- and preference-based traversal of the car park with dynamic preference.

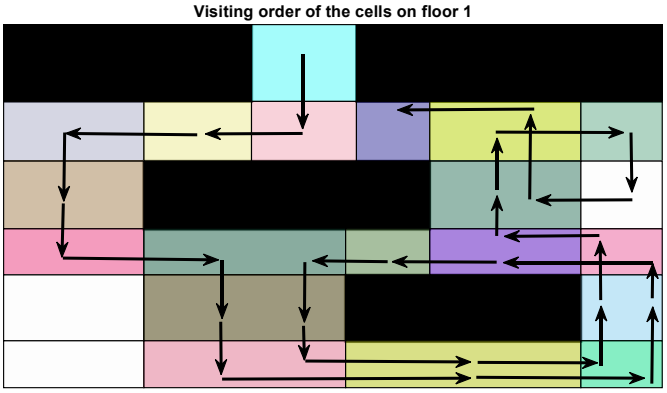


Figure 16. The visitedness- and preference-based traversal of the car park with dynamic preference while avoiding unnecessary cells.

### 5.3. Visitedness- and preference-based traversal with dynamic preference

This method differs from the previous one by applying dynamic preference. This means the preference value of the cell visited last is decreased during the traversal. In this case, the preference values of the cells decreased by 2 every time they were visited.

As a result, every cell could be visited; the preference values of the cells visited first were decreased enough so that the algorithm would select the cells with lower preference values as the next cells. Figure 15 provides an example of this method.

### 5.4. Visitedness- and preference-based traversal with dynamic preference while avoiding unnecessary cells

A map may contain cells that are not adjacent to parking spaces, as they do not contain any $c \in C_{vis}$ points, so it is not necessary to visit them. In this case, the preference values of these cells can be changed to 0, and they can be marked as visited at the beginning. The algorithm will then only visit these cells in order to reach other cells. In Figure 16, the white cells represent cells that do not need to be visited.

### 5.5. Preference- and visitedness-based traversal

This method selects the next cell based on its preference value (see Figure 17). If a number of neighbouring cells have the same preference value, the algorithm selects the following one based on the visitedness of the cell (see Figure 18).

As can be seen in Figure 19, the cells with the highest preference values are visited multiple times, while cells with lower preference values may remain unvisited.

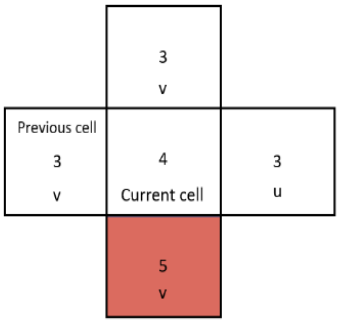


Figure 17. The preference value determines the following cell.

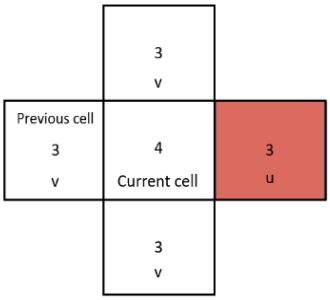


Figure 18. The visitedness of the cells determines the following cell.

### 5.6. Preference- and visitedness-based traversal with additional preference values

As preference- and visitedness-based traversal mainly selects the next cell based on its preference value, this method should be used in cases where the driver wants to park at a given location. Additional preference values can also be applied to attract the traversal to the desired location.

Figure 20 provides an example of the applied additional preference values. The highest additional preference value (5) was added to the cell that contains the preferred location (see Figure 10), so each neighbouring cell gets one smaller additional preference value until the fourth neighbouring cell (the range of the preference is 5).

It can be seen in Figure 21 that the algorithm navigates to the highly preferred area in the shortest possible path, and subsequently, the traversal only moves around one obstacle by traversing the cells with the highest preference values.

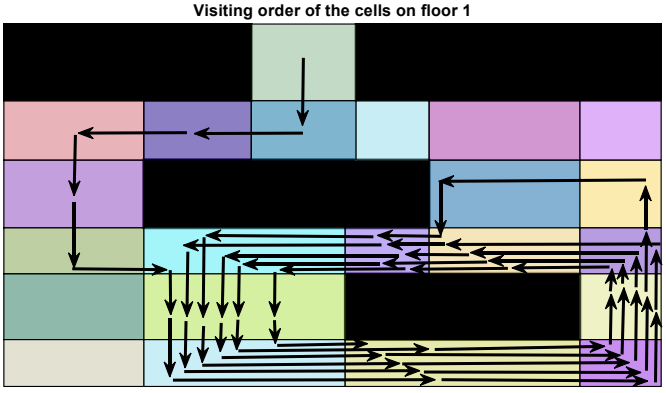


Figure 19. Preference- and visitedness-based traversal.

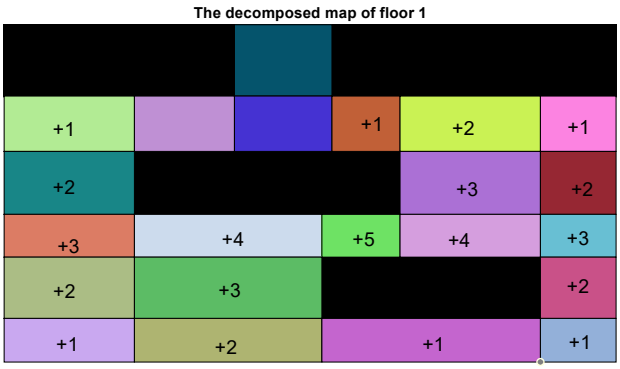


Figure 20. The additional preference values.

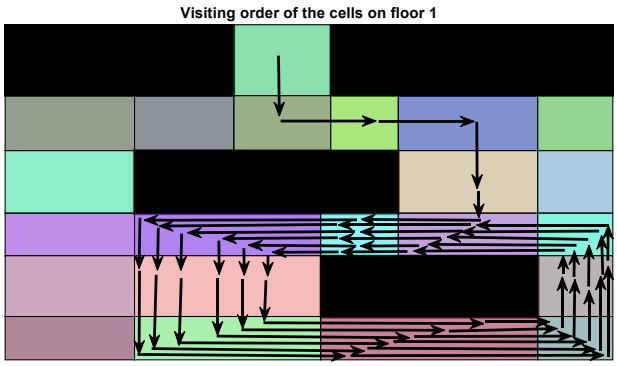


Figure 21. Preference- and visitedness-based traversal with additional preference values.

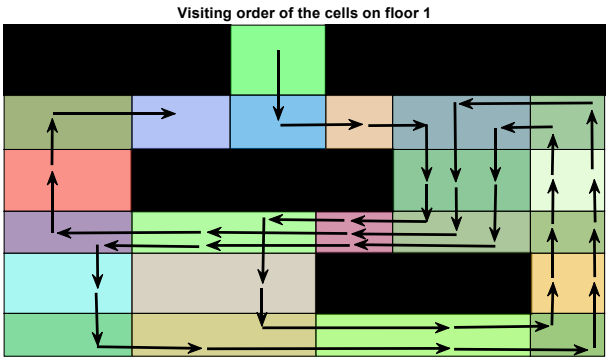


Figure 22. Preference- and visitedness-based traversal with additional preference values and dynamic preference.

### 5.7. Preference- and visitedness-based traversal with additional preference values and dynamic preference

The previous algorithm can also be applied when using dynamic preference values. In this case, the preference values of the visited cells are decreased by 2 every time they are visited. It can be seen in Figure 22 that the algorithm visits the cells with high preference values, but then also visits the neighbouring cells. In this case, every cell is visited.

This method might be the closest to human thinking. First, it searches for free parking spaces in the highly preferred area, then it goes a bit further and returns to the preferred location. Finally, it also visits the cells that are furthest from the preferred cells.

### 5.8. Making the traversal repeatable

The presented traversals do not lead back to the initial position. It is possible that no free parking spaces were found during the first traversal of the car park, so the traversal should be repeated in order to find a suitable free parking space. There are two possible solutions to this problem: regenerate the traversal from the current cell as the initial cell or plan a path back to the initial cell from the current cell so that the original traversal can be repeated.

## 6. COST OF THE TRAVERSAL

The quality of the detected free parking space can be measured with a cost function. There are two main factors that should be considered: the time spent searching for a free parking space (which is proportional to the driven-route length $w_1$ in Section 3) and the distance from the preferred location ($w_2$ in Section 3). There is a trade-off between these two factors, but the quality of a parking space also depends on personal preferences: the importance of the proximity to a preferred location or a short driven-route length.

The $RouteLength$ ($w_1$ in Section 3) (5) is the sum of the distances between the cell centres during the traversal, the $Nearness$ ($w_2$ in Section 3) (6) to the preferred location is the preference ($prefVal$) weighted sum of the Euclidean distances measured from the preferred location and the cost function ($cost$), which is calculated for each cell, is a personal preference ($\alpha$) weighted sum of the $RouteLength$ and $Nearness$ (7). The smaller the $\alpha$ value, the better a closer parking space.

$$RouteLength = \sum_{i=1}^{N} \text{Length}(Route_i), \tag{5}$$

$$Nearness = \sum_{i=1}^{P} \frac{prefVal_i}{\sum_{j=1}^{P} prefVal_j} \cdot \text{dist}(position, prefLoc_i), \tag{6}$$

$$cost = \alpha \cdot RouteLength + (1-\alpha) \cdot Nearness, \tag{7}$$

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

The quality of a free parking space can be determined by the value of the cost function. The driver should establish a threshold value, and free parking spaces at lower costs are adequate. The better a parking space is, the lower the value of the cost function.

## 7. COMPARISON OF THE TRAVERSALS

In this section, the previously presented methods are compared based on the step number (Section 7.1), the cell visitedness ratio (Section 7.2), the number of visits to each cell (Section 7.3) and the cost function (Section 7.4).

### 7.1. Step number

The step number gives the number of steps needed to visit every cell at least once. Traversing from one cell to another is considered as one step. As the algorithm stops when more than a given number of steps is exceeded, the maximum number of steps in this map is 57.

Table 1 shows how many steps are needed when different methods were applied. It can be seen that when the visitedness- and preference-based method (Sections 5.2–5.4) was applied, fewer unvisited cells remained. When there were cells marked as unnecessary to visit (Section 5.4), only 30 steps were sufficient to visit all the other cells, and in the end, two out of three marked

Table 1. The number of steps needed when applying different methods.

| Method section number | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
|---|---|---|---|---|---|---|
| Number of steps | 57 | 56 | 30 | 57 | 57 | 42 |
| Number of unvisited cells | 1 | 0 | 2 | 5 | 8 | 0 |

cells remained unvisited. When dynamic preference (Sections 5.3, 5.4 and 5.7) was applied, fewer steps were needed and fewer cells remained unvisited.

### 7.2. Cell visitedness ratio

The cell visitedness ratio represents the number of visited cells relative to the number of cells in each step. Figure 23 shows the visitedness ratio for the different methods using different colours. Cells marked as unnecessary in Section 5.4 are also included in the number of cells. The legend gives the section number in which the traversal method is presented.

First, every cell is visited for the first time, so this ratio grows at each step. When applying the preference- and visitedness-based method without dynamic preference (Sections 5.5 and 5.6), the final value of the ratio is lower than in other cases. These methods visit only the cells with high preference values (60 %–75 % of the cells). The other methods visit nearly all the cells (more than 80 % of the cells). Applying dynamic preference (Section 5.7) makes the visiting of the remaining unvisited cells more probable. When visitedness- and preference-based traversal (Sections 5.2–5.4) was used, the algorithm visited the unvisited neighbouring cells first, so there were fewer cells that remained unvisited.

### 7.3. The number of visits to each cell

The following diagrams show the number of visits to each cell. Numbers on the horizontal axis represent the indices of the cells shown in Figure 9.

#### Visitedness- and preference-based traversal

This method's traversal (Section 5.2) visits the unvisited cells first. Cells with high preference values are visited more frequently (5–6 times), but there is only one cell (cell 14) that is unvisited. For example, the preference value of cell 13 is 9 (see orange bar in Figure 24), and this cell was visited 6 times (see blue bar in Figure 24). The diagram demonstrating the number of visits to each cell can be seen in Figure 24.

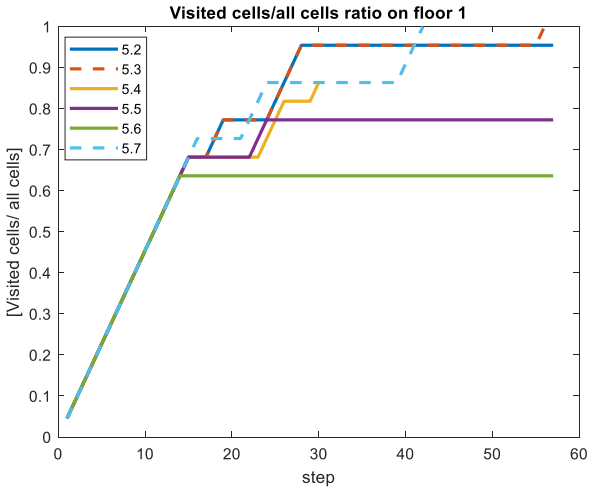


Figure 23. Cell visitedness ratio of the presented methods.

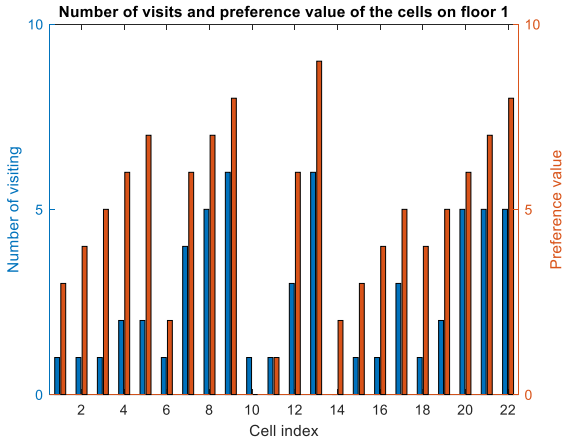


Figure 24. Number of visits to each cell (method presented in Section 5.2).

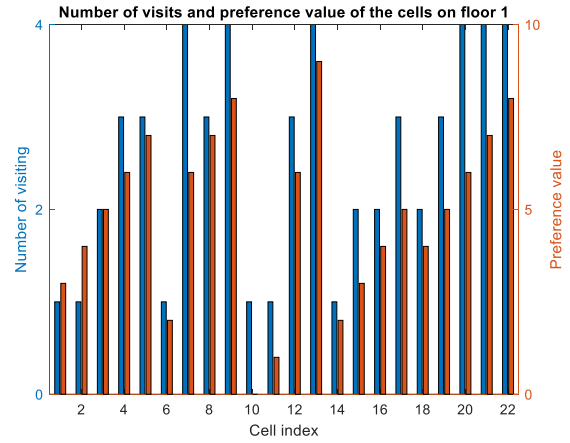


Figure 25. Number of visits to each cell (method presented in Section 5.3); the preference values shown in this figure are the original values, not the decreased values.

#### Visitedness- and preference-based traversal with dynamic preference

By applying dynamic preference (Section 5.3), all the cells become visited and the maximum number of visits to each cell is four. Most of the cells are visited once or twice, so the application of dynamic preference decreases the number of visits (Figure 25).

#### Visitedness- and preference-based traversal with dynamic preference while avoiding unnecessary cells

There can be cells in a map that do not need to be visited because there are no parking spaces around them. In this case, they are only visited if the traversal goes through them in order to reach another cell (Section 5.4). In Figure 26, the only unvisited cells are unnecessary ones (unnecessary cell indices: 4, 5, 19). Due to dynamic preference, the other cells are visited only once or twice.

#### Preference- and visitedness-based traversal

This method (Section 5.5) visits the cells with a high preference value first and more frequently because the algorithm selects the next cell based on the preference value of the neighbouring cells. In Figure 27, it can be seen that most of the visited cells are visited 5–6 times, but there are a large number of unvisited cells.

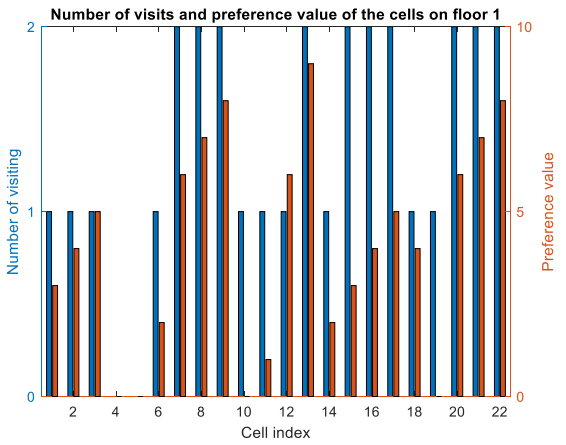


Figure 26. Number of visits to each cell (method presented in Section 5.4); the preference values shown in this figure are the original values, not the decreased values.

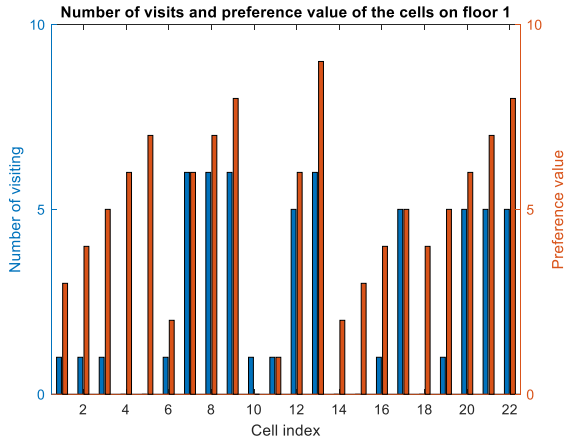


Figure 27. Number of visits to each cell (method presented in Section 5.5).

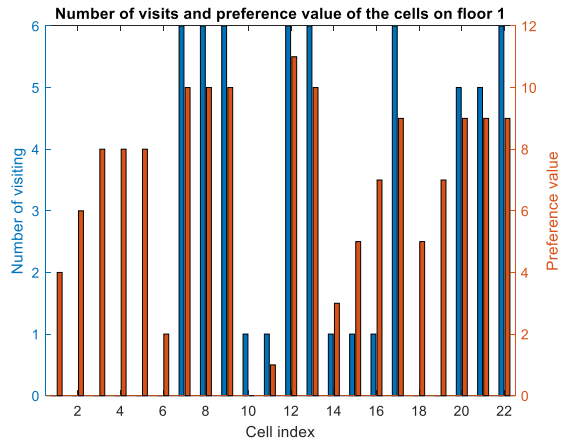


Figure 28. Number of visits to each cell (method presented in Section 5.6).

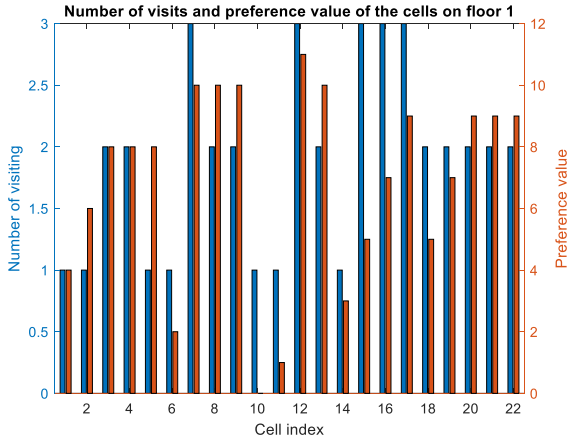


Figure 29. Number of visits to each cell (method presented in Section 5.7); the preference values shown in this figure are the original values, not the decreased values.

#### Preference- and visitedness-based traversal with additional preference values

In Figure 28, it can be seen that due to the application of additional preference values fewer cells are visited, but they are visited 6 times. This algorithm (Section 5.6) navigates to the most preferred location (there was an additional preference applied with a range of 5 and a value of 5 in the cell with index 12) in the shortest possible route (these cells are only visited once), then it only moves around the preferred area.

#### Preference- and visitedness-based traversal with additional preference values and dynamic preference

When applying dynamic preference (Section 5.7), all the cells are visited, and each cell is visited a maximum of 3 times instead of 5–6 times (Figure 29).

### 7.4. Cost function

The algorithm can decide whether a free parking space is suitable or not based on a cost function. The defined cost function is based on two factors: the driven-route length and distance from the preferred location.

The estimated route length of a method depends on the step number; the higher the step number, the longer the route length (see Table 2). The route length is measured in pixels (the real distance depends on the graphic scale of the map).

The other aspect of cost function is the distance from the preferred location. The distance is calculated at every step, and it depends strongly on the traversal method. The minimum distance from the preferred location is 50 pixels in this example.

#### Visitedness- and preference-based traversal

This method (Section 5.2) visits the unvisited cells first, only reaching the possible minimum distance 3 times during the traversal. It can be seen in Figure 30 that the distance is between 180 and 330 pixels most of the time.

#### Visitedness- and preference-based traversal with dynamic preference

Figure 31 shows that the shape of the distance function is similar to the function in Figure 30. This function also has three minimum points due to dynamic preference (Section 5.3); only the visiting order of the cells is different, and the unvisited cells are also visited.

Table 2. The route length of each method.

| **Method section number** | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
|---|---|---|---|---|---|---|
| **Number of steps** | 57 | 56 | 30 | 57 | 57 | 42 |
| **Route length** | 6752 | 6608 | 3184 | 6555 | 6671 | 4379 |

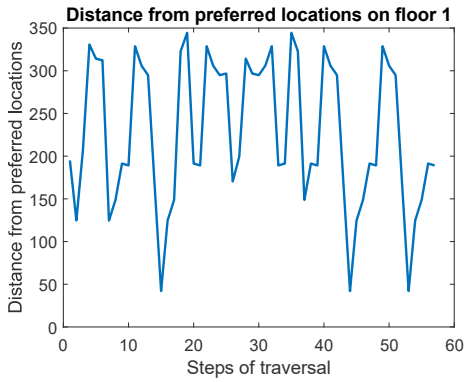


Figure 30. Distance from preferred locations (method presented in Section 5.2).

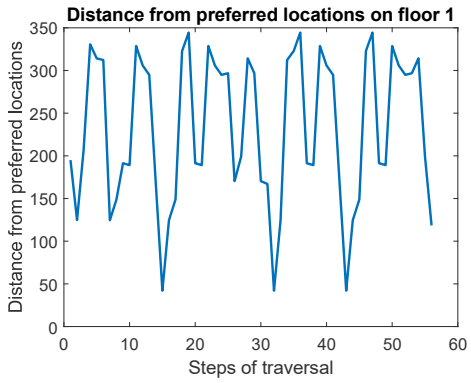


Figure 31. Distance from preferred locations (method presented in Section 5.3).

#### Visitedness- and preference-based traversal with dynamic preference while avoiding unnecessary cells

This method (Section 5.4) avoids the cells that have no parking spaces next to them, so the step number is lower than in the previous methods. It can be seen in Figure 32 that the traversal visits the cell nearest to the preferred location only once, but the length of this traversal is much shorter due to the smaller number of steps.

#### Preference- and visitedness-based traversal

This method (Section 5.5) visits cells with a high preference value more often. The preferred location is far from the initial cell, so it has a high distance value. Without an additional preference value, the algorithm also visits the preferred location 5 times, as can be seen in Figure 33.

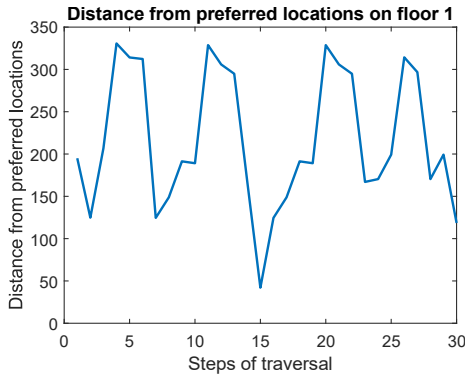


Figure 32. Distance from preferred locations (method presented in Section 5.4).

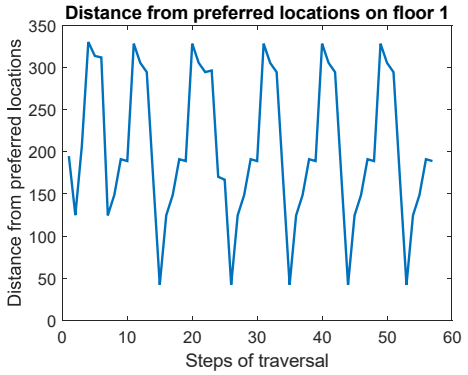


Figure 33. Distance from preferred locations (method presented in Section 5.5).

#### Preference- and visitedness-based traversal with additional preference values

By applying additional preference values (Section 5.6), the traversal reaches the minimal distance from the preferred location 6 times (Figure 34). It can also be seen that the traversal repeats the same path, as one part of the function is repeated 5 times.

#### Preference- and visitedness-based traversal with additional preference values and dynamic preference

As a result of dynamic preference (Section 5.7), the cells that are further from the preferred location are also visited. The traversal has only three minimum points (Figure 35) because when a cell becomes visited, the preference value of this cell is decreased.

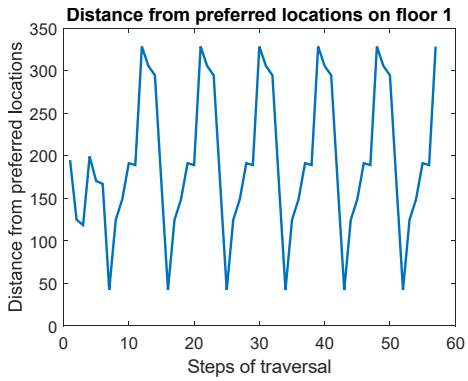


Figure 34. Distance from preferred locations (method presented in Section 5.6).

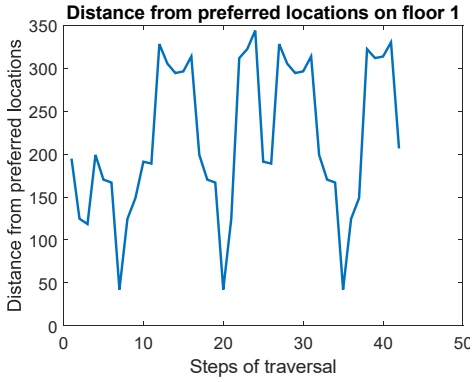


Figure 35. Distance from preferred locations (method presented in Section 5.7).

## 8. HANDLING MULTI-STOREY CAR PARKS

As there is a considerable lack of free parking spaces in city centres, an increasing number of multi-storey car parks have been constructed to ensure there are enough free parking spaces to meet demand. The traversal of these car parks is similar to those in the presented methods.

The storeys of a multi-storey car park can be handled individually. This means that the traversal of each floor should be planned independently from the other floors, with the only extra requirement being the minimising of the transition between levels. To plan a traversal of each floor, the map of the floor must be known. Some floors are preferred over others, so these floors should be traversed first. If there is no free parking space on the preferred floor, the driver must either go around again or go to another floor.

The traversal to another floor can be forced so that the preference values of all the cells can be set to zero except for the cell representing the ramp to the other floor, which has a high preference value with a large range. Figure 36 shows the map of the first floor of a car park, and Figure 37 shows its second floor. The car park entrance and the ramp to the next floor are also indicated on the maps. The maps of the floors are the same, with only different entrance and exit locations.

The planned traversals for the whole car park can be seen in Figure 38 to Figure 40. There are no additional preference values during the traversal, and the traversal is planned based on visitedness and preference, with dynamic preference values. The traversal of the first floor (Figure 38) is much longer than the traversal of the second floor (Figure 40). The first floor can be traversed in 56 steps, but the second floor needs only 33 steps to be traversed. The traversal method of the floors is the same, only the entrance locations are different. This example shows how the traversal depends on the location of the entrance point.

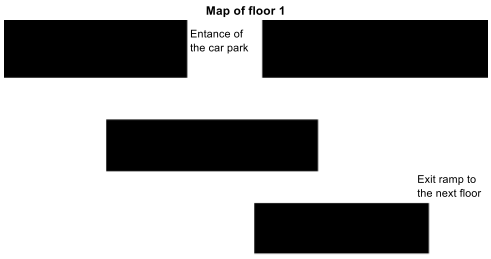


Figure 36. Map of the first floor.

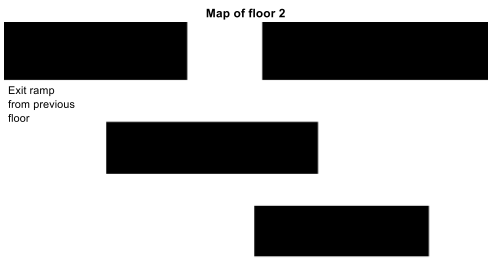


Figure 37. Map of the second floor.

## 9. CONCLUSION

As searching for a free parking space is a time-consuming task, the aim of this paper is to design different car park exploration strategies.

The implemented algorithms used the core concepts of CPP algorithms, which is possible because the car park exploration problem is similar to CPP problems. CPP algorithms are used to plan the paths of vacuum-cleaner robots, lawnmower robots and robots for different purposes, which are designed to reach every free point of a configuration in the shortest possible time while avoiding obstacles. During car park exploration, the vehicle does not have to reach every free point of the map, it only has to drive by all the possible free parking spaces.

The car park map is decomposed by using trapezoidal cell decomposition. This method leads to cells with large areas, and the planned traversal contains reversals. If the map is decomposed using both $x$ and $y$ axes, the created cells are smaller, and every cell has only one neighbouring cell in each direction. In this case, the traversal can take personal preferences

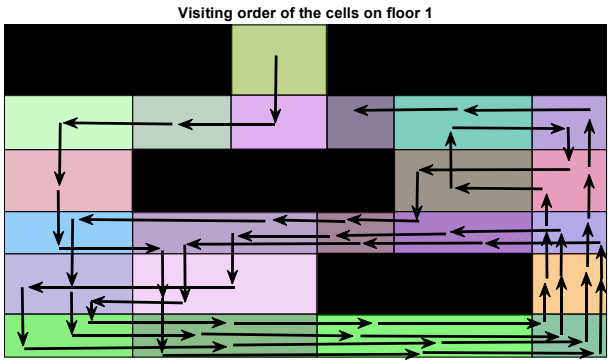


Figure 38. Traversal of the first floor.

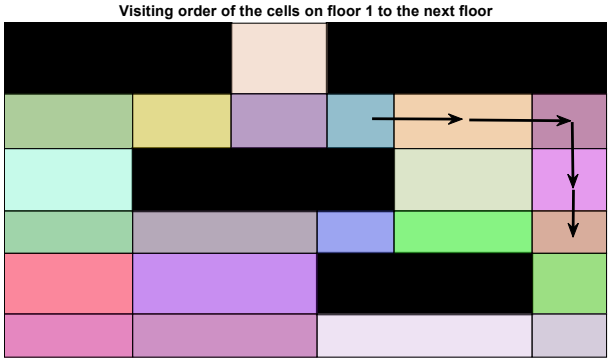


Figure 39. Traversal of the first floor to the exit ramp.

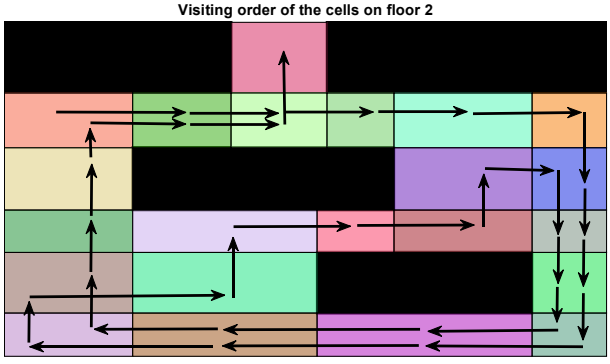


Figure 40. Traversal of the second floor.

into account by using the wavefront algorithm. The distance values can be modified in order to attract the traversal to a given location (e.g. entrances, lifts, etc.).

The traversal can be planned by taking the preference values and the visitedness of the cells into account. The first method presented only takes the preference values into account, so the traversal does not visit every cell on the map, only the ones with high preference values. Another method chooses the next cell based on the preference value, but in case of equal preference values, the next cell is an unvisited one. The third wavefront algorithm-based traversal is based on visitedness and preference, and it visits the unvisited neighbouring cells first. This method is more likely to visit all the cells on the map.

In order to compare the presented methods, quality characteristics were defined: step number, the cell visitedness ratio at each step, the number of visits to each cell and a cost function. The step number shows how many steps are needed in order to visit every cell; the algorithm stops if the step number exceeds a given number. The cell visitedness ratio shows the ratio of the visited cells at each step. The cost function is based on two factors: the driven-route length and the weighted sum of the distance from the preferred locations. If a free parking space is found, the decision as to whether it is suitable is based on the cost function.

The implemented algorithms were tested in simulations, the results of which are detailed in Section 5. The simulation results demonstrate that the different methods can be used in different situations. The visitedness- and preference-based traversals visit nearly every cell on the map. If dynamic preference is applied, there is a higher chance that every cell becomes visited. It is also possible that there are cells that do not need to be visited. In this case, their preference value can be changed to 0, and they are marked as visited at the beginning. These cells only become visited when the traversal passes through them to reach other cells. Visitedness- and preference-based methods should be used when it is important to find a free parking space as soon as possible. These methods visit the cells with high preference values more frequently. If additional preference values are applied, the traversal moves around the preferred area. If dynamic preference is applied, the traversal visits the preferred cells first, then goes further away from the preferred area. Preference- and visitedness-based methods should be used if it is important to park near the preferred location.

Future work will include testing the implemented methods in a real environment and handling situations in which multiple vehicles are searching for free parking spaces at the same time.

## ACKNOWLEDGEMENT

The research reported in this paper and carried out at the Budapest University of Technology and Economics has been supported by the National Research Development and Innovation Fund (TKP2020 Institution Excellence Subprogram, Grant No. BME-IE-MIFM) based on the charter issued by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology.

## REFERENCES

[1] Faheem Zafari, S. A. Mahmud, G. M. Khan, M. Rahman, H. Zafar, A survey of intelligent car parking system, J. of Applied Research and Technology 11 (2013) pp. 714-726.
DOI: 10.1016/S1665-6423(13)71580-3

[2] F. Al-Turjman, A. Malekloo, Smart parking in IoT-enabled cities: a survey, Sustainable Cities and Society, Volume 49 (2019), pp. 2210-6707.
DOI: 10.1016/j.scs.2019.101608

[3] A. B. Ádám, L. Kocsány, E. G. Szádeczky-Kardoss, V. Tihanyi. Parking lot exploration strategy, Proc. of the 19th IEEE Int. Symp. on Computational Intelligence and Informatics and the 7th IEEE Int. Conf. on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo), Szeged, Hungary, 14-16 November 2019, pp. 000169-000174.
DOI: 10.1109/CINTI-MACRo49179.2019.9105160

[4] A. Athira, S. Lekshmi, P. Vijayan, B. Kurian, Smart parking system based on optical character recognition, Proc. of the 3rd Int. Conf. on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23-25 April 2019, pp. 1184-1188.
DOI: 10.1109/ICOEI.2019.8862517

[5] Parkl Digital Technologies Kft, Parkl innovative parking, 2020. Online [Accessed 15 September 2021]
https://parkl.net/hu/

[6] Smart Lynx Kft, Parker, 2020. Online [Accessed 15 September 2021]
https://smartlynx.hu/

[7] E. Szádeczky-Kardoss, B. Kiss, Designing a tracking controller for passenger cars with steering input, Period. Polytech. Elec. Eng. 52 (2008) pp. 137-144.
DOI: 10.3311/pp.ee.2008-3-4.02

[8] E. Szádeczky-Kardoss, B. Kiss, Continuous-curvature paths for mobile robots, Period. Polytech. Elec. Eng. 53 (2009) pp. 63-72.
DOI: 10.3311/pp.ee.2009-1-2.08

[9] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous Systems 61 (2013) pp. 1258-1276.
DOI: 10.1016/j.robot.2013.09.004

[10] R. Bellman, Dynamic programming treatment of the travelling salesman problem, J. ACM 9 (1962) pp. 61-63.
DOI: 10.1145/321105.321111

[11] S. Kumar Ghosh, Approximation algorithms for art gallery problems in polygons and terrains, in: WALCOM: Algorithms and Computation. M. S. Rahman, S. Fujita (editors). Springer, Berlin, Heidelberg, 2010, ISBN 0302-9743, pp. 21-34.

[12] W. P. Chin, S. Ntafos, Optimum watchman routes, Information Processing Letters 28 (1988) pp. 39-44.
DOI: 10.1016/0020-0190(88)90141-X

[13] H. Choset, E. Acar, A. A. Rizzi, J. Luntz, Exact cellular decompositions in terms of critical points of morse functions, Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, USA, 24-28 April 2000, pp. 2270-2277.
DOI: 10.1109/ROBOT.2000.846365

[14] M. A. Akkus, Trapezoidal cell decomposition and coverage, Middle East Technical University, Department of Computer Engineering. Online [Accessed 14 September 2021]
https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng786/hw3.html

[15] H. Choset, Coverage of known spaces: the Boustrophedon cellular decomposition, Auton. Robots 9 (2000) pp. 247-253.
DOI: 10.1023/A:1008958800904

[16] S. Raghavan, Distributed algorithms for hierarchical area coverage using teams of homogeneous robots. 11 2020, Master's thesis. Indian Institute Of Technology Madras.

[17] J. Park, Cell decomposition course: introduction to autonomous mobile robotics, Intelligent Systems and Robotics Lab. Division of Electronic Engineering, Chonbuk National University. Online [Accessed 14 September 2021]
https://cupdf.com/document/cell-decomposition-course-introduction-to-autonomous-mobile-robotics-prof.html

[18] A. Das, M. Diu, N. Mathew, C. Scharfenberger, J. Servos, A. Wong, J. Zelek, D. Clausi, S. Waslander, Mapping, planning, and sample detection strategies for autonomous exploration, J. of Field Robotics 31 (2014), pp. 75-106.
DOI: 10.1002/rob.21490

[19] M. Mcnally, Walking the grid, Robotics 52 (2006), Pages 151–155. Online [Accessed 20 September 2021] https://dl.acm.org/doi/pdf/10.5555/1151869.1151889

[20] A. B. Ádám, L. Kocsány, E. G. Szádeczky-Kardoss, Cell decomposition based parking lot exploration, Proc. of the Workshop on the Advances of Information Technology, Budapest, Hungary, 30 January 2020, pp. 5-12.

[21] Y. Gabriely, E. Rimon, Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot, Proc. of the IEEE Int. Conf. on Robotics and Automation, Washington, DC, USA, 11-15 May 2002, pp. 954-960. DOI: 10.1109/ROBOT.2002.1013479

[22] Y. Gabriely, E. Rimon, Competitive on-line coverage of grid environments by a mobile robot, Computational Geometry 24 (2003) pp. 197-224. DOI: 10.1016/S0925-7721(02)00110-4