

Robot Coverage Path Planning Based on Iterative Structured Orientation

Ernő Horváth^a, Claudiu Pozna^{a, b}, Radu-Emil Precup^c

herno@sze.hu, cp@unitbv.ro, radu.precup@upt.ro

^a Department of Computer Engineering, Széchenyi István University,
Egyetem tér 1, 9026 Győr, Hungary

^b Department of Automation and Information Technology, Transilvania University
of Brasov, Str. Mihai Viteazu 5, Corp V, et. 3, 500174 Brasov, Romania

^c Department of Automation and Applied Informatics, Politehnica University of
Timisoara, Bd. V. Parvan 2, 300223 Timisoara, Romania

Abstract: Coverage path planning for mobile robots aims to compute the shortest path that ensures the overlap of a given area, with applications in various domains. This paper proposes a coverage path planning strategy, referred to as Iterative Structured Orientation Coverage, which has two main advantages over the state-of-the-art, namely it is versatile and it is capable to handle complex environments. The path planning strategy is expressed as three new approaches to coverage path planning. The suggested approaches are validated by simulation and experimental results. The source codes along with the test set are available in a public repository.

Keywords: Auxiliary lines; Coverage path planning; Iterative Structure Orientation Coverage; Main lines; Mobile

1 Introduction

As shown in [1] and [2], robot coverage path planning (CPP) deals with the problem of covering a certain area with a movable object as, for example, with a mobile robot (MR). CPP makes use of two classes [3]: it is complete if it guarantees complete coverage or heuristic in other cases. There are also two main CPP strategies: offline if there is an a priori known map, otherwise online if the robot needs to discover the environment.

The most widely used approaches to CPP are Random Path Planning (RPP) [4], Exact Cellular Decomposition (ECD) [1], Boustrophedon Cellular DeComposition (BCDC) [1] [38], Backtracking Spiral Algorithm (BSA) [4], Internal Spiral Search

(ISS) [5], U-turn A* Path Planning (UAPP) [5] and Neural Network (NN)-based CPP [6]. A critical analysis of these approaches is presented as follows.

RPP imposes the MR to move on several random trajectories. Each time when the current trajectories are obstructed a new one is chosen and next repeated. RPP is a simple but not effective approach. Combining RPP with preprogrammed trajectory patterns such as spirals and serpentine and/or a wall following mechanism the algorithm may increase efficiency [4]. Due to its simplicity, the path planning algorithms specific to RPP are used in nowadays popular autonomous robotic vacuum cleaners.

ECD uses cells to fill the whole map. Usually the MR covers each cell using simple back-and-forth motions. After the current cell is covered, the robot moves to another cell. Finally the whole map will be covered. The Trapezoidal Decomposition is a particular case of ECD given in [1], and characterized by the decomposition of the map into trapezoids, which are covered with simple back-and-forth motions.

BCDC has been introduced in [4] and the path planning algorithms based on BCDC became popular as highlighted in [5]. The word boustrophedon literally means “the way of the ox” in Greek [4]. The original supposition is that the map is composed from polygons, so it is a line map [6]. BCDC exploits this hypothesis and generates cells (easy to be covered) and finally generates the connection between these cells. BCDC performs an exact cellular decomposition, and each cell in the boustrophedon is covered with back and forth motions [4].

The drawback of the polygonal decomposition is the big number of cells. This problem has been corrected by merging cells [7]. Fig. 1 illustrates an example of BCDC-based solution where four cells are produced. The cells are generated using a beam of parallel lines because of intersection with obstacles. These cells are actually convex polygons that do not have any holes. Consequently, the cells are covered with back and forth motions. This aspect is also illustrated in Fig. 1. Finally, the thick path shows how each cell is connected to ensure a complete coverage.

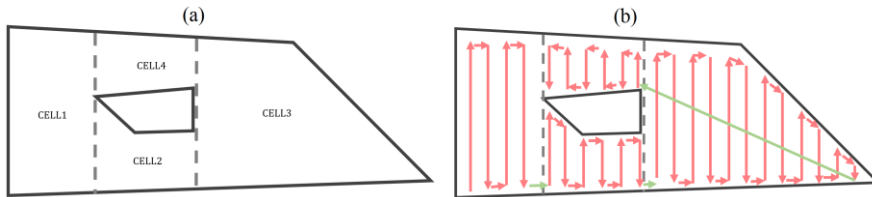


Figure 1
The cells (a) and the path (b) generated by BCDC

BSA is similar to ISS. The MR traverses the area in a certain direction. If the area is free, i.e., not covered, the MR will move forward. If it is already covered or there is an obstacle in the way, the MR will turn perpendicularly [5].

UAPP combines the heuristic feature of A* algorithm with the U-turn search algorithm [5]. The MR moves from the origin point using a U-turn algorithm, and next plans the shortest path in term of an A* algorithm [5].

The NN-based approach generates collision-free complete coverage paths in known environments by producing shunting equations [6]. Several features specific to NNs that offer convenient input-output maps to model complex systems [7–13] can be included and combined with the NN-based CPP approach.

Some recent applications of CPP are reported in [14–18]. The path planning approaches can be applied to various categories and applications of MRs [19–27].

This paper proposes a novel CPP strategy, which is referred to as Iterative Structured Orientation Coverage (ISOC). ISOC uses discrete grid maps and targets the complete coverage. The specific feature of our approach is the combination of two ideas, considering the whole area as one unit and using the BCDC-based motion.

The ISOC approach uses the concept of main lines. These main lines are actually a beam of parallel lines, which have a particular orientation in the map. This orientation ensures a set of straight lines with maximum length, surrounded by the map and interrupted by the obstacles. By composing (or linking) these lines we obtain the minimum length path. The composition stage relates our solution with optimization problems as, for example, the traveling salesman problem (TSP) or other problems in different applications treated with classical or evolutionary-based algorithms [9, 11, 28–36], and also represents the advantage of the ISOC approach with respect to the state-of-the-art reported in [1–6, 14–19].

Three solutions to obtain the main lines are proposed. These solutions are inserted in the ISOC strategy resulting in three new ISOC approaches.

The paper is organized as follow: the main contribution of the paper, which is the ISOC approaches, is discussed in the next section and presented in a unified formulation. Section 3 deals with the validation of the proposed approaches by simulation and experimental results for the Khepera III differential drive robot, and a comparative study is included. Section 4 is dedicated to concluding and summarizing remarks and to outlining the future research directions.

2 Iterative Structured Orientation Coverage Approaches

As mentioned in Section 1, the map (domain) coverage in ISOC uses the main lines concept. The main lines represent a beam of parallel lines, which are trimmed by the map boundary in segments. The final path is obtained by composing these segments with additional segments. The additional segments ensure the path continuity. The main lines have a particular orientation, which ensures the minimum length of the path, i.e., the goal to make the MR navigate on long straight lines. Fig. 2 illustrates this concept by means of Fig. 2 (a) that illustrates the map and the main lines and Fig. 2 (b) that illustrates the main segments (lines) and the auxiliary segments.

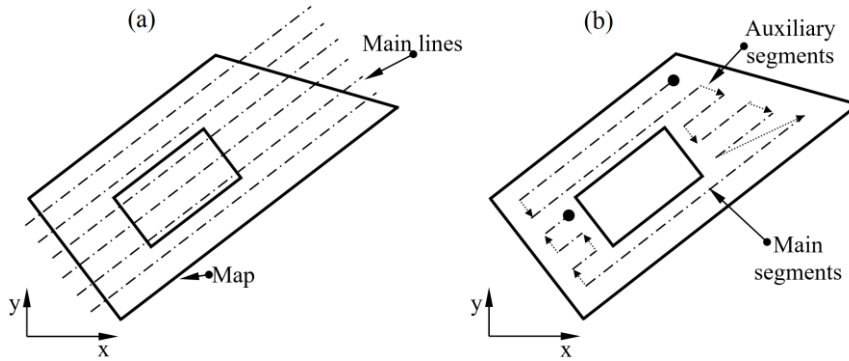


Figure 2

The map and the main lines (a), the main segments (lines) and the auxiliary ones (b)

The initial data of the CPP approach is the map. The map is generated from a picture and modeled by the map matrix $M = [M_{ij}]_{i=1\dots n, j=1\dots m} \in \mathbb{R}^{n \times m}$ with the elements

$$M_{ij} = \begin{cases} 1, & \text{if pixel } (i, j) \text{ is black,} \\ 0, & \text{if pixel } (i, j) \text{ is white,} \end{cases} \quad (1)$$

where n is the number of horizontal pixels and m is the number of vertical pixels.

Using the main lines concept, the problem of finding a minimum length path, which covers the whole map, reduces to the following steps, 1, 2 and 3:

1. Find the appropriate main line, i.e., the orientation of the beam of parallel lines.
2. Obtain the main segments.
3. Link these segments with auxiliary segments such that the final (continuous) path has a minimum length.

These steps will be described in the next sub-sections, and next organized as a unified algorithm specific to the ISOC approaches.

2.1 Finding the Main Lines (the Orientation of the Beam of Parallel Lines)

The equation of the beam of parallel lines expressed in the discrete domain is

$$\begin{cases} L_{q_1 k_1} \equiv i = \left\lfloor j \frac{q_1}{(m+1)} \right\rfloor + k_1 + 1, & \text{if } \alpha \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right], \\ L_{q_2 k_2} j = \left\lfloor i \frac{q_2}{(n+1)} \right\rfloor + k_2 + 1, & \text{if } \alpha \in \left[-\frac{\pi}{2}, -\frac{\pi}{4}\right) \cup \left(\frac{\pi}{4}, \frac{\pi}{2}\right] \end{cases} \quad (2)$$

where α is the line slope in the continuous domain, $\alpha = \tan^{-1}\left(\frac{q_1}{m+1}\right)$ or $\alpha = \tan^{-1}\left(\frac{q_2}{n+1}\right)$, $q_1 = 1 \dots m$ or $q_2 = 1 \dots n$ have a direct effect on the slope in the discrete domain, with the unified notation $q \in \{q_1, q_2\}$, $\lfloor x \rfloor$ indicates generally the integer part of $x \in \mathfrak{R}$, $k_1 = \beta_1 \delta_1$ or $k_2 = \beta_2 \delta_2$ is the intercept with the unified notation $k \in \{k_1, k_2\}$ for both horizontal and vertical axes, $\beta_1 = 0 \dots (n-1)/\delta_1$, $\beta_2 = 0 \dots (m-1)/\delta_2$ the integer steps δ_1 and δ_2 are computed in terms of

$$\delta_1 = \left\lfloor b \frac{\sqrt{q_1^2 + (m+1)^2}}{m+1} \right\rfloor, \quad \delta_2 = \left\lfloor b \frac{\sqrt{q_2^2 + (n+1)^2}}{n+1} \right\rfloor, \quad (3)$$

b is the distance between the lines (the robot width), $L_{q_1 k_1}$ and $L_{q_2 k_2}$ are the lines that belong to the beam with the unified notation $L_{qk} \in \{L_{q_1 k_1}, L_{q_2 k_2}\}$ for both axes. Fig. 3 illustrates several examples of lines for different slopes.

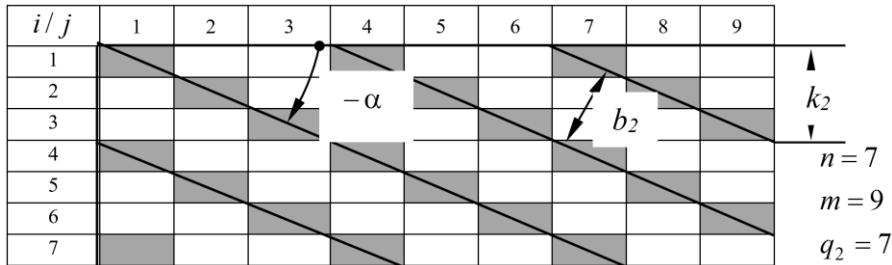


Figure 3
Examples of lines in the discrete domain

Each line can be associated with a matrix $\Lambda^{qk} = [\Lambda_{ij}^{qk}]_{i=1 \dots n, j=1 \dots m} \in \mathfrak{R}^{n \times m}$ with the elements Λ_{ij}^{qk}

$$\Lambda_{ij}^{qk} = \begin{cases} 1, & \text{if } (i, j) = (L_{q_1 k_1}, j) \wedge (i, j) = (i, L_{q_2 k_2}), \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Three solutions for the computation of the main lines are proposed in this paper. The first two solutions generate a beam of lines and define the main segments as the intersection between the map and the beam of lines, compute the segments length and compute (or approximate) the path length in terms of the sum of these lengths. An iterative process is conducted to compute the slope of the beam of lines, which generates a set of path lengths. The result, represented the main segments, is related to the minimum length path as the solution to the optimization problem

$$q^* = \underset{q \in D_q}{\operatorname{argmin}} \sum_{k \in D_k} \Gamma(\Lambda^{qk}, M) \quad (5)$$

where q^* gives the optimum slope in the discrete domain, D_q is the discrete domain of slope, D_k is the intercept domain, the general notation $\Gamma(\Lambda^{qk}, M)$ is the general notation for the path length:

$$\Gamma(\Lambda^{qk}, M) = \lambda \sum_{i=1}^n \sum_{j=1}^m p_{ij}, \quad p_{ij} = \begin{cases} 0, & \text{if } M_{ij} = 1, \\ 1, & \text{if } \Lambda_{ij}^{qk} = 1, \end{cases} \quad (6)$$

where the general notation $\lambda \in \{\lambda_1, \lambda_2\}$ is used for the distance between the points calculated as

$$\begin{aligned} \lambda_1 &= \sqrt{\frac{(m+1)^2 + q_1^2}{m+1}}, \\ \lambda_2 &= \sqrt{\frac{(n+1)^2 + q_2^2}{n+1}}. \end{aligned} \quad (7)$$

The first solution consists of the following steps:

Step 1.1. The lines expressed in (2), which depend on q and k , are generated.

Step 1.2. The matrices Λ^{qk} with the elements Λ_{ij}^{qk} expressed in (4) are generated.

Step 1.3. The path length $\Gamma(\Lambda^{qk}, M)$ is computed according to (6).

Step 1.4. The objective function in (5) is computed in terms of the sum $\sum_{k \in D_k} \Gamma(\Lambda^{qk}, M)$ for $q = \text{const}$ and variable k , $k \in D_k$.

Step 1.5. The optimization problem defined in (5) is solved considering that the objective function in the right-hand term of (4) depends on the variable q , $q \in D_q$, and the solution to this optimization problem, i.e., the variable that gives the minimum path length, is q^* .

The first two solutions differ by the slope domain and by the map definition. The first solution preserves the initial map and defines a continuous domain of slope $D = [0, \pi]$ in order to include all possible orientations of the beam of parallel lines. Fig. 4 illustrates an example of beam of parallel lines used in the first solution.

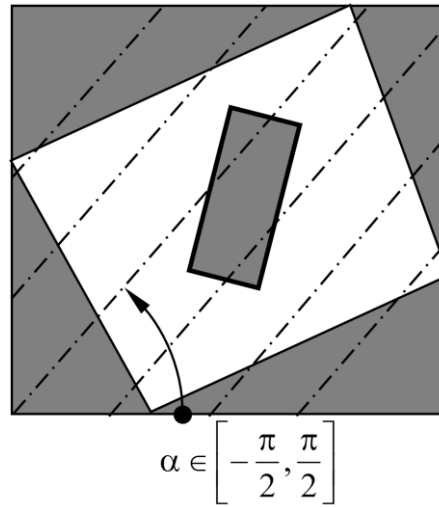


Figure 4

Example of beam of parallel lines used in the first solution

The second solution defines a new map using a composition of the initial map and uses a smaller domain of slope, i.e. $D = [0, \pi/4]$. Fig. 5 exemplifies a beam of parallel lines used in the second solution.

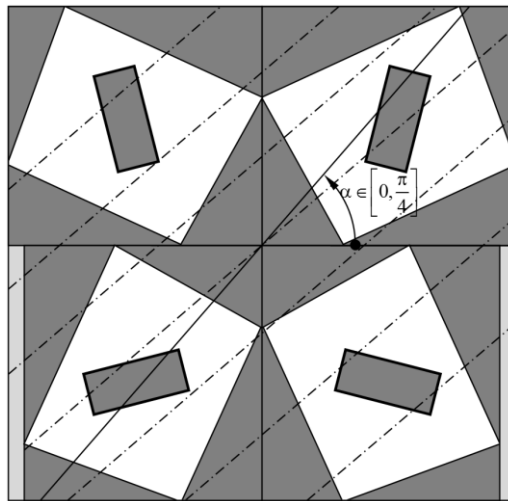


Figure 5

Example of beam of parallel lines used in the second solution

The second solution is based on the generation of a new map by the union of four maps that are rotated as shown in Fig. 5. These four maps correspond to the four quadrants I, II, III and IV, obtained as follows.

The map in the quadrant I is $M_I = [M_{I,ij}]_{i=1\dots n, j=1\dots m} \in \mathbb{R}^{n \times m}$, with the map matrix elements:

$$M_{I,ij} = M_{ij}, i = 1\dots n, j = 1\dots m \quad (8)$$

The map in the quadrant II is $M_{II} = [M_{II,ij}]_{i=1\dots n, j=1\dots m} \in \mathbb{R}^{n \times m}$, with the map matrix elements:

$$M_{II,ij} = M_{im-j+1}, i = 1\dots n, j = 1\dots m \quad (9)$$

The map in the quadrant III is M_{III} , obtained in terms of the composition

$$M_{III} = [P|M^T] \in \mathbb{R}^{m \times m}, P = [P_{ij}]_{i=1\dots m, j=1\dots m-n}, P_{ij} = 1 \quad (10)$$

where the subscript T indicates matrix transposition.

The map in the quadrant IV is $M_{IV} = [M_{IV,ij}]_{i=1\dots n, j=1\dots m} \in \mathbb{R}^{n \times m}$, with the map matrix elements:

$$M_{IV,ij} = M_{in-j+1}, i = 1\dots n, j = 1\dots m \quad (11)$$

The second solution consists of the following steps:

Step 2.1. The map matrix in the four quadrants is computed using (8) to (11).

Steps 2.2 to 2.6. These are the steps 1.1 to 1.5 in the first solution.

The third solution approximates the main lines with the map axis, which is inspired from the properties specific to mechanical inertia. The map axis slope is obtained in terms of

$$\alpha = \frac{1}{2} \tan^{-1} \left(\frac{2I_{xy}}{I_y - I_x} \right) \quad (12)$$

where the following center of gravity-type relationships are employed:

$$\begin{aligned} I_{xy} &= \sum_{i=1}^n \sum_{j=1}^m i_c j_c M_{ij}, \quad I_x = \sum_{i=1}^n \sum_{j=1}^m j_c^2 M_{ij}, \quad I_y = \sum_{i=1}^n \sum_{j=1}^m i_c^2 M_{ij} \\ x &= \frac{\sum_{i=1}^n \sum_{j=1}^m j M_{ij}}{\sum_{i=1}^n \sum_{j=1}^m M_{ij}}, \quad y = \frac{\sum_{i=1}^n \sum_{j=1}^m i M_{ij}}{\sum_{i=1}^n \sum_{j=1}^m M_{ij}} \\ i_c &= i - y, \quad j_c = j - x, \quad i = 1\dots n, \quad j = 1\dots m \end{aligned} \quad (13)$$

and M_{ij} are the elements of the map matrix M defined in (1). The beam of parallel lines is next computed using (2).

The third solution consists of the following steps:

Step 3.1. The map axis slope is computed using (12) and (13).

Steps 3.2 to 3.6. These are the steps 1.1 to 1.5 in the first solution.

An example of application of the third solution is given in Fig. 6.

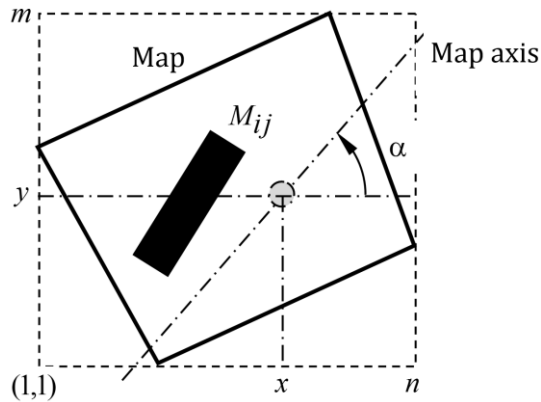


Figure 6

Example of map and map axis in used in the third solution

2.2 Finding the Auxiliary Lines

The result of the three solutions presented in the previous sub-section is an ordered beam of segments. The lines order has been defined in the process of definition of the main lines using the intercept of each line expressed in (2). The definition of the main segments splits the lines in several segments producing a list of segments ordered (in their turn) using a left to right convention. This means that each segment has two kinds of neighborhoods, i.e., the segments that belong to the lists of neighbor main lines and the segments that do not belong to the same list. The second type of segments is eluded because of the obstacles between these segments. An ordered beam of segments is illustrated in Fig. 7.

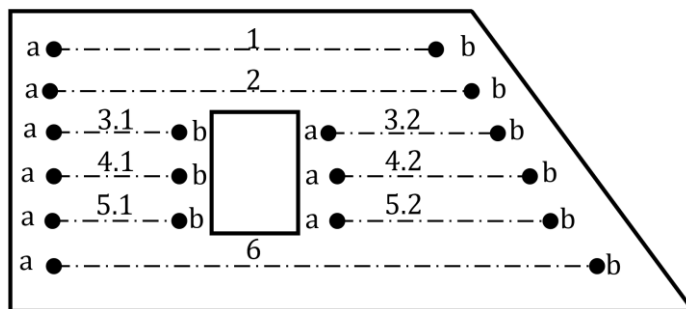


Figure 7

Example of ordered beam of segments

Each segment defines two nodes, a and b . The graph G is defined between the nodes of the main segments:

$$G = (V, E), \quad (14)$$

where the set of nodes (vertices) is V :

$$V = \{i.j.k | i = 1 \dots n_s, j = 1 \dots n_i, k = a \text{ or } b\} \quad (15)$$

n_s is the number of main segments, and n_i is the number segments generated from i^{th} main line, and the set of edges is E :

$$E(i.j.k, p.r.l) = \begin{cases} 1 & \text{if } p = i + 1 \text{ or } i.j = p.r, \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

Equation (16) evaluates the existence of the edge between two nodes. The edge exists if $E(i.j.k, p.r.l) = 1$, that means if either $p = i + 1$, i.e. between the segments of two successive lines (the lines are not skipped) or $i.j = p.r$, i.e. between the points of the same segment.

This idea assigns a segment to a node. At the first glance each node can be connected with any other node. In order to avoid this complexity, a heuristic is proposed in this paper to connect only neighbor segments.

The graph related to the segments illustrated in Fig. 7 is presented in Fig. 8 (a).

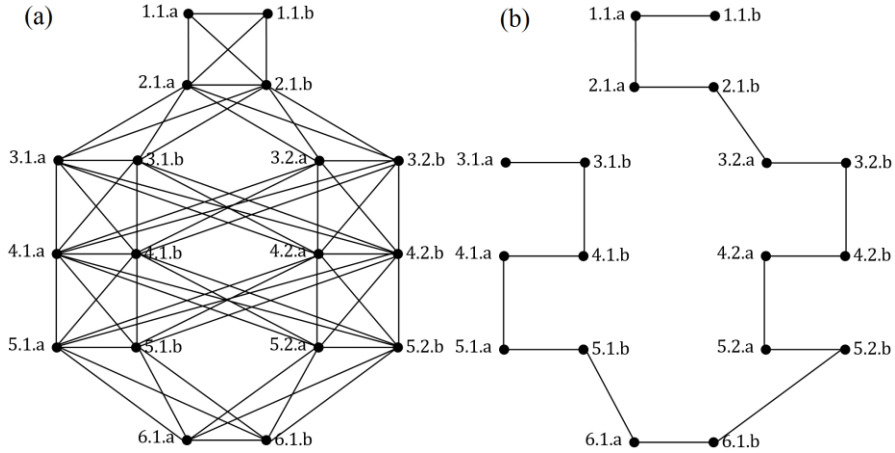


Figure 8

The graph (a) and its solution (b)

Each edge of the graph is associated to a cost function. The simplest cost function definition is the distance between the nodes. If the nodes can be linked with a straight line, the computation of the distance is simple. Contrarily, if obstacles interfere, a trajectory between nodes must be defined. The graph can be further simplified by trimming these edges in order to fulfill the objective to minimize the cost function, i.e., to minimize the path length.

The problem of visiting all segments (nodes) is related to the TSP [32]. In contrast to the classical TSP approach, which assumes that all of them need to be visited once, the ISOC approach proposed in this paper does not have this constraint. The only constraint imposed here is that after a node is reached it is mandatory to visit the second node of the segment so the covering of the main segments is ensured. Fig. 8 (b) points out a possible solution that starts with the node 1.1.b and ends with the node 3.1.a.

Concluding, the auxiliary segments are the lines (or trajectories) that connect the main segments and ensure a minimum path length.

2.3 The Algorithm Specific to the Iterative Structured Orientation Coverage Approaches

Using the steps 1, 2 and 3 related to finding a minimum length path specified at the beginning of Section 2 and the two previous sub-sections, the algorithm specific to the ISOC approaches to robot CPP is referred to as the ISOC algorithm and consists of the following steps:

Step A. The initial data regarding the robot CPP is provided in terms of:

- The map (1), i.e., a discrete representation of the boundaries and the obstacles
- The overall dimensions of the robot

Step B. The main lines are defined and the main segments are found using one of the three solutions presented in Sub-section 2.1, which lead to the minimum length path as the solution to the optimization problem defined in (5), expressed as the optimum slope q^* of the main segments.

Step C. The auxiliary lines are obtained using the results presented in sub-section 2.2 by:

- Computing the visiting order between the main segments
- Computing the path between the main segments using the graph G defined in (14), (15) and (16)

Concluding, the three ISOC approaches are characterized in a unified presentation by the ISOC algorithm. The difference between the three approaches is in the step B, where one of the three solutions to obtain the minimum length path presented in Sub-section 2.1 is included, and leads to one of the three new ISOC approaches. The ISOC approach with the first solution to obtain the minimum length path is next referred to as the first ISOC approach, the ISOC approach with the second solution to obtain the minimum length path is next referred to as the second ISOC approach, and the ISOC approach with the third solution to obtain the minimum length path is next referred to as the third ISOC approach.

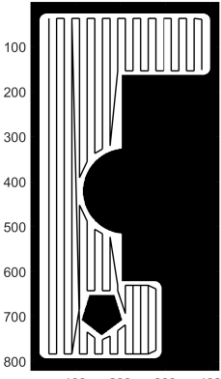
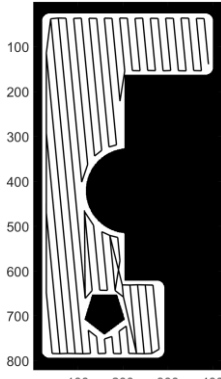
3 Validation by Simulation and Experiments

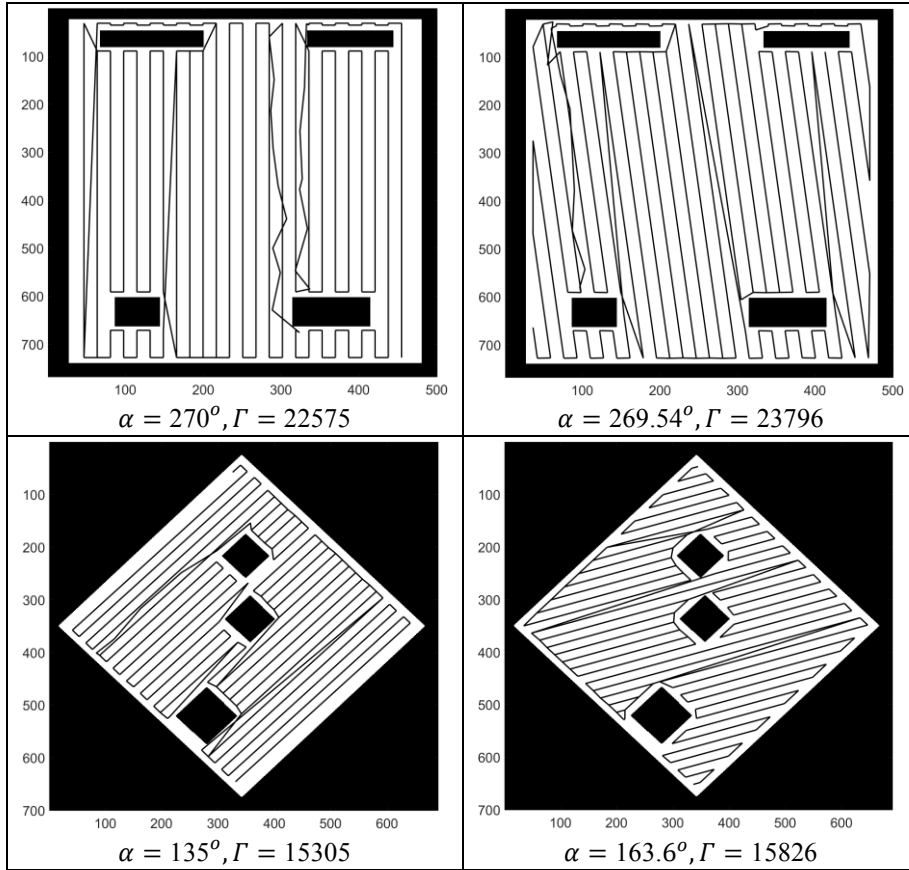
This section validates the ISOC approaches presented in the previous section by simulations and experiments conducted on mobile robots. The validation by simulation includes a comparison between the proposed approaches, and of the proposed approaches with another well-known approach discussed in Section 1, namely the BCDC approach. The validation by experiments is focused on a Khepera mobile robot.

3.1 Simulation Results

The comparison between the ISOC approaches has been carried out for artificial (generated) maps. The results are illustrated in Table 1. Although the first two ISOC approaches use optimal solutions and the third ISOC approach is heuristic, the results are close. As shown in the first row, the third ISOC approach gives a better result because of the discretization errors.

Table 1
Comparison between the proposed approaches

Results using the first two approaches	Results using the third approach
 $\alpha = 90^\circ, \Gamma = 14443$	 $\alpha = 84.52^\circ, \Gamma = 13684$



Three types of maps have been used to validate the ISOC algorithm presented in the previous section: a simulated map obtained from V-REP [2], a real-life measurement map [37], and an artificially generated map. The maps developed in V-REP and imported to Matlab are called simulation maps. The real-world measurement maps are available datasets, which we have been downloaded from [37]. These maps represent the third floor common area of the MIT Stata Center (Dreyfoos Center) [38]. The artificially generated maps have been obtained by either hand drawing or randomly using Matlab. These maps are illustrated in Fig. 9 as follows: three artificial maps in Fig. 9 (a), (b) and (d), and a real-world map, i.e. the test bench taken from our laboratory snapshot in Fig. 9 (c).

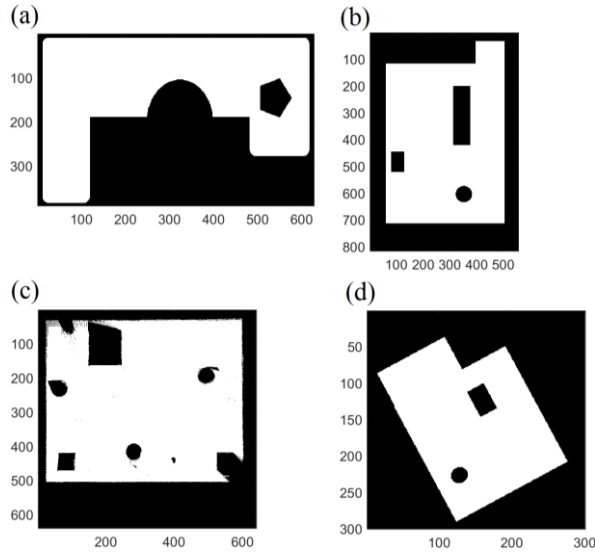


Figure 9

Real-world and artificially generated maps

The proposed approaches are next compared to the BCDC approach, and the results are presented in Table 2. The comparison shows that the ISOC algorithm is always slower than the BCDC algorithm, but most of the time it generates a shorter path. This confirms the results that have been foreseen at the very beginning of the creation of the ISOC approaches and algorithm: ISOC deals with complex maps better, meaning that it generates shorter paths, but this is reflected in its increased computational complexity.

Table 2

Comparison between the proposed approaches and the BCDC approach

Map size (pixel)	Number of holes	Occupied ratio (%)	BCDC-based path length (mm)	ISOC-based path length (mm)	BCDC-based computation time (s)	ISOC-based computation time (s)	Image
435600	1	73.04	9984.41	9625.15	15.65	32.17	Fig. 9 (a)
453696	3	55.77	21392.96	18449.71	10.79	45.65	Fig. 9 (b)
409600	5	60.06	27214.68	22278.05	14.82	96.97	Fig. 9 (c)
90000	2	39.23	2879.35	2880.04	10.54	49.48	Fig. 9 (d)

The comparison was done with an i7 processor, clock rate of 3.7 GHz and 16 GB memory. The comparison of the three proposed approaches from the point of view of the computation time shows that the first approach is the most time consuming, and the second approach reduces the computation time if a parallel computing (for each quadrant) is considered and the approximation is made faster.

Future research will focus on performance enhancement of the ISOC algorithm by means of four approaches. First, the parallelization of certain sections of code will be carried out by means of a GPGPU-based approach. Second, the use of abstract data types as lists, trees or graphs will be also considered. Third, own targeted functions will be written instead of the used general-purpose functions of an API or a toolbox. And last, the generated path can be executed with a help of a nonlinear model and controller [40-43].

Acknowledgement

This work was supported by a grant of the Executive Agency for Higher Education, Research, Development and Innovation Funding (UEFISCDI), project number PN-II-RU-TE-2014-4-0207, by grants from the Partnerships in priority areas – PN II program of UEFISCDI, project numbers PN-II-PT-PCCA-2013-4-0544 and PN-II-PT-PCCA-2013-4-0070, by the Széchenyi István University, and by the TAMOP - 4.2.2.C-11/1/KONV-2012-0012 Project "Smarter Transport - IT for co-operative transport system".

References

- [1] H. Choset, "Coverage for robotics – A survey of recent results, *Annals of Mathematics and Artificial Intelligence*" Vol. 31, No. 1, pp. 113-126, 2001
- [2] M. Freese, S. Singh, F. Ozaki, N. Matsuhira, Virtual Robot Experimentation Platform V-REP: A versatile 3D robot simulator, in: N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, O. von Stryk (Eds.), *Simulation, Modeling, and Programming for Autonomous Robots*, Springer-Verlag, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6472, pp. 51-62, 2010
- [3] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, *Robotics and Autonomous Systems* Vol. 61, No. 12, pp. 1258-1276, 2013
- [4] M. Waanders, Coverage path planning for mobile cleaning robots, in: *Proc. 15th Twente Student Conference on IT*, Twente, The Netherlands, pp. 1-10, 2011
- [5] Z.-Y. Cai, S.-X. Li, Y. Gan, R. Zhang, Q.-K. Zhang, Research on complete coverage path planning algorithms based on A* algorithms, *The Open Cybernetics & Systemics Journal* Vol. 8, pp. 418-426, 2014
- [6] S. X. Yang, C. Luo, A neural network approach to complete coverage path planning, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* Vol. 34, No. 1, pp. 718-724, 2004
- [7] I. Škrjanc, S. Blažič, O. E. Agamennoni, Identification of dynamical systems with a robust interval fuzzy model, *Automatica* Vol. 41, No. 2, pp. 327-332, 2005
- [8] A. G. Martin, R. E. Haber Guerra, Internal Model Control Based on a Neurofuzzy System for Network Applications. A Case Study on the High-

- Performance Drilling Process, IEEE Transactions on Automation Science and Engineering, Vol. 6, No. 2, pp. 367-372, 2009
- [9] J. Vaščák, M. Paľa, Adaptation of fuzzy cognitive maps for navigation purposes by migration algorithms, International Journal of Artificial Intelligence Vol. 8, No. 12, pp. 20-37, 2012
- [10] R.-E. Precup, R.-C. David, E. M. Petriu, Grey Wolf Optimizer Algorithm-Based Tuning of Fuzzy Control Systems With Reduced Parametric Sensitivity, IEEE Transactions on Industrial Electronics, Vol. 64, No. 1, pp. 527-534, 2017
- [11] R.-E. Precup, P. Angelov, B. S. J. Costa, M. Sayed-Mouchaweh, An overview on fault diagnosis and nature-inspired optimal control of industrial process applications, Computers in Industry Vol. 74, pp. 75-94, 2015
- [12] T. T. Mac, C. Copot, D. T. Tran, R. De Keyser, Heuristic approaches in robot path planning: A survey, Robotics and Autonomous Systems Vol. 86, pp. 13-28, 2016
- [13] K. Sasaki, K. Noda, T. Ogata, Visual motor integration of robot's drawing behavior using recurrent neural network, Robotics and Autonomous Systems Vol. 86, pp. 184-195, 2016
- [14] J. Conesa-Muñoz, G. Pajares, A. Ribeiro, Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment, Expert Systems with Applications Vol. 54, pp. 364-378, 2016
- [15] I. A. Hameed, A. la Cour-Harbo, O. L. Osen, Side-to-side 3D coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths, Robotics and Autonomous Systems Vol. 76, pp. 36-45, 2016
- [16] M. Torres, D. A. Pelta, J. L. Verdegay, J. C. Torres, Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction, Expert Systems with Applications Vol. 55, pp. 441-451, 2016
- [17] D. S. Li, X. L. Wang, T. Sun, Energy-optimal coverage path planning on topographic map for environment survey with unmanned aerial vehicles, Electronics Letters Vol. 52, No. 9, pp. 699-701, 2016
- [18] J. Granna, I. S. S. Godage, R. Wirz, K. D. Weaver, R. J. Webster, J. Burgner-Kahrs, A 3-D volume coverage path planning algorithm with application to intracerebral hemorrhage evacuation, IEEE Robotics and Automation Letters Vol. 1, No. 2, pp. 876-883, 2016
- [19] J. K. Tar, I. J. Rudas, J. F. Bitó, Group theoretical approach in using canonical transformations and symplectic geometry in the control of approximately modelled mechanical systems interacting with an unmodelled environment, Robotica Vol. 15, pp. 163-179, 1997
- [20] J. K. Tar, I. J. Rudas, K. R. Kozłowski, Fixed point transformations-based approach in adaptive control of smooth systems, in: K. R. Kozłowski (Ed.),

- Robot Motion and Control 2007, Springer-Verlag, London, Lecture Notes in Control and Information Sciences, Vol. 360, pp. 157-166, 2007
- [21] S. Blažič, A novel trajectory-tracking control law for wheeled mobile robots, *Robotics and Autonomous Systems* Vol. 59, No. 11, pp. 1001-1007, 2011
- [22] M. Arndt, S. Wille, L. de Souza, V. Fortes Rey, N. Wehn, K. Berns, Performance evaluation of ambient services by combining robotic frameworks and a smart environment platform, *Robotics and Autonomous Systems* Vol. 61, No. 11, pp. 1173-1185, 2013
- [23] K. Bolla, Zs. Cs. Johanyák, T. Kovács, G. Fazekas, Local center of gravity based gathering algorithm for fat robots, in: L. T. Kóczy, C. R. Pozna, J. Kacprzyk (Eds.), *Issues and Challenges of Intelligent Systems and Computational Intelligence*, Springer-Verlag, Berlin, Heidelberg, *Studies in Computational Intelligence*, Vol. 530, pp. 175-183, 2014
- [24] J. Vaščák, N. H. Reyes, Use and perspectives of fuzzy cognitive maps in robotics, in: E. I. Papageorgiou (Ed.), *Fuzzy Cognitive Maps for Applied Sciences and Engineering: From Fundamentals to Extensions and Learning Algorithms*, Springer-Verlag, Berlin, Heidelberg, *Intelligent Systems Reference Library*, Vol. 54, pp. 253-266, 2014
- [25] M. Reichardt, T. Föhst, K. Berns, An overview on framework design for autonomous robots, *Information Technology* Vol. 57, No. 2, pp. 75-84, 2015
- [26] C. Pozna, R.-E. Precup, P. Földesi, A novel pose estimation algorithm for robotic navigation, *Robotics and Autonomous Systems* Vol. 63, pp. 10-21, 2015
- [27] B. Kovács, G. Szayer, F. Tajti, M. Burdelis, P. Korondi, A novel potential field method for path planning of mobile robots by adapting animal motion attributes, *Robotics and Autonomous Systems* Vol. 82, pp. 24-34, 2016
- [28] F. G. Filip, Decision support and control for large-scale complex systems, *Annual Reviews in Control* Vol. 32, No. 1, pp. 61-70, 2008
- [29] J.-S. Chiou, S.-H. Tsai, M.-T. Liu, A PSO-based adaptive fuzzy PID-controllers, *Simulation Modelling Practice and Theory* Vol. 26, pp. 49-59, 2012
- [30] D. Wijayasekara, O. Linda, M. Manic, C. G. Rieger, FN-DFE: Fuzzy-Neural Data Fusion Engine for enhanced resilient state-awareness of hybrid energy systems, *IEEE Transactions on Cybernetics* Vol. 44, No. 11, pp. 2065-2075, 2014
- [31] R.-E. Precup, M.-C. Sabau, E. M. Petriu, Nature-inspired optimal tuning of input membership functions of Takagi-Sugeno-Kang fuzzy models for anti-lock braking systems, *Applied Soft Computing* Vol. 27, pp. 575-589, 2015
- [32] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, A. Masegosa, A. Perallos, Good Practice Proposal for the Implementation, Presentation, and

- Comparison of Metaheuristics for Solving Routing Problems, *Neurocomputing*, Vol. 271, pp. 2-8, 2018
- [33] S. B. Ghosn, F. Drouby, H. M. Harmanani, A parallel genetic algorithm for the open-shop scheduling problem using deterministic and random moves, *International Journal of Artificial Intelligence* Vol. 14, No. 1, pp. 130-144, 2016
- [34] D. Azar, K. Fayad, C. Daoud, A combined ant colony optimization and simulated annealing algorithm to assess stability and fault-proneness of classes based on internal software quality attributes, *International Journal of Artificial Intelligence* Vol. 12, No. 2, pp. 137-156, 2016
- [35] P. Cabrera-Guerrero, A. Moltedo-Perfetti, E. Cabrera, F. Paredes, Comparing two heuristic local search algorithms for a complex routing problem, *Studies in Informatics and Control* Vol. 25, No. 4, pp. 411-420, 2016
- [36] F. Gaxiola, P. Melin, F. Valdez, J. R. Castro, O. Castillo, Optimization of type-2 fuzzy weights in backpropagation learning for neural networks using GAs and PSO, *Applied Soft Computing* Vol. 38, pp. 860-871, 2016
- [38] H. Choset, P. Pignon, Coverage path planning: The boustrophedon decomposition, in: A. Zelinsky (Ed.), *Field and Service Robotics*, Springer-Verlag, London, pp. 203-209, 2008
- [39] Maps of common areas of 3rd floor Stata Center (Dreyfoos Section) provided by P. E. Missiuro, posted at <http://cres.usc.edu/radishrepository/view-one.php?name=mitstata3rdfloordreyfoos> and accessed in March 2017
- [40] C. Pozna, F. Troester, R.-E. Precup, J. K. Tar, S. Preitl, On the Design of an Obstacle Avoiding Trajectory: Method and Simulation, *Mathematics and Computers in Simulation*, Vol. 79, No. 7, pp. 2211-2226, 2009
- [41] R.-E. Precup, M. L. Tomescu, S. Preitl, E. M. Petriu, J. Fodor, C. Pozna, Stability Analysis and Design of a Class of MIMO Fuzzy Control Systems, *Journal of Intelligent and Fuzzy Systems*, Vol. 25, No. 1, pp. 145-155, 2013
- [42] J. K. Tar, J. F. Bitó, I. J. Rudas, Contradiction Resolution in the Adaptive Control of Underactuated Mechanical Systems Evading the Framework of Optimal Controllers, *Acta Polytechnica Hungarica*, Vol. 13, No. 1, pp. 97-121, 2016
- [43] A. Ürmös, Z. Farkas, M. Farkas, T. Sándor, L. T. Kóczy, A. Nemcsics, Application of Self-Organizing Maps for Technological Support of Droplet Epitaxy, *Acta Polytechnica Hungarica*, Vol. 14, No. 4, pp. 207-224, 2017