IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

Towards a continuous solution of the d-visibility watchman route problem in a polygon with holes

Jan Mikula[1,2] and Miroslav Kulich[2]

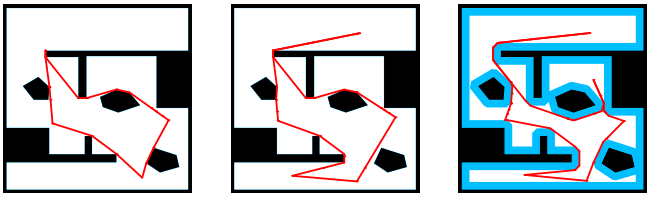
***Abstract*—A new heuristic solution framework is proposed to address the challenging *watchman route problem* (WRP) in a polygonal domain, which can be viewed as an offline version of the robot exploration task. The solution is the shortest route from which the robot can visually inspect a known 2D environment. Our framework considers a circular robot with radius r equipped with an omnidirectional sensor with limited visibility range d. Instead of a standard *decoupled* solution, the framework generates a set of specifically constrained regions covering the domain and then solves the *traveling salesman problem with continuous neighborhoods* (TSPN) to obtain the solution route. The TSPN is solved by another proposed heuristic algorithm that finds a discretized solution first and then improves it back in the continuous domain. The whole framework is evaluated experimentally, compared to two approaches from the literature, and shown to provide the highest-quality solutions. The current version of the framework is one step from a fully continuous approach to the WRP that we will address in the future.**

***Index Terms*—Task and Motion Planning; Motion and Path Planning; Computational Geometry; Watchman Route Problem; Traveling Salesman Problem with Neighborhoods.**

I. Introduction

A classic problem in computational geometry and robotics is the *watchman route problem* (WRP). Assume a mobile agent (e.g., watchman or autonomous robot) with the ability to see a certain portion of the environment around itself. The WRP is the problem of finding the shortest closed collision-free route to be traversed by the agent such that it sees the whole environment in which it operates. Unlike the *exploration task*, which unravels *online*, the WRP is set *offline*, i.e., the map of the environment is known in advance.

Problem statement: We focus on a WRP variant suited for efficient visual inspection of a known environment performed by an autonomous mobile robot. We assume that the environment has a 2D representation, such as a floor-plan, which can be turned into a polygona domain, i.e., a simple polygon with holes, denoted as $\mathcal{W}$. The robot is assumed to be holonomic, circular with radius r, and equipped with an omnidirectional sensor with limited visibility range d, placed in its center. The goal is to plan the shortest polyline route the robot could follow to see the whole environment. We say route τ is *collision-free* in $\mathcal{W}$ if it lies entirely inside $\mathcal{W}$ and its minimal distance from any segment of $\mathcal{W}$ is no smaller than r. We say that route τ *visually covers* $\mathcal{W}$, or just *covers* $\mathcal{W}$, if all points of $\mathcal{W}$ are *d-visible* from at least one point along τ. Two points are *d-visible* to each other if the line segment between them lies entirely in $\mathcal{W}$ and its length is no bigger than d. *The problem is to find the shortest closed route τ that is collision-free in $\mathcal{W}$ and visually covers $\mathcal{W}$.* We denote the definition above as the rd-WRP, or d-WRP if $r=0$, or *classic WRP* if $r=0$ and $d=\infty$ (see Fig. 1 for examples). The *classic WRP* and d-WRP are known to be NP-hard for polygons with holes [1]–[3].



classic WRP l=44.4 | d-WRP d=7.0, l=74.9 | rd-WRP r=0.85, d=7.0, l=88.4

Fig. 1: Examples of the WRP: (20 m × 20 m) environment $\mathcal{W}$ is shown in white, obstacles are black, unreachable parts of $\mathcal{W}$ are blue, and the solution route is red. The value of l is the solution's length, and d, r, and l are in meters.

A standard solution approach to this kind of problem is *decoupling* [4], which splits the original problem into two subproblems that are solved independently: 1) the *art gallery problem* (AGP): find a minimal set of discrete sensing locations (guards) that completely *cover* the environment, and 2) the *traveling salesman problem* (TSP): determine the order of guards' visits such that the length of the final route is minimal. However, despite both parts being solved optimally, the optimal solution to the WRP is not guaranteed because *decoupling* independently minimizes the cost of sensing and the cost of motion, while the WRP wants to minimize a combination of both costs [5]. As a result, *decoupling* does not always provide solutions of sufficient quality. Another existing approach is using a soft-computing technique called *self-organizing map* (SOM) [6], [7]. The SOM creates an initial route *covering* only a part of the environment. This partial solution is then iteratively modified so that its parts are

Manuscript received November 10, 2021; revised February 12, 2022; accepted March 6, 2022.

This paper was recommended for publication by Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000470) and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS21/185/OHK3/3T/37.

[1] Jan Mikula is with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, 121 35 Praha 2, Czech Republic, mikulj14@fel.cvut.cz.

[2] Jan Mikula and Miroslav Kulich are with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Jugoslávských partyzánů 1580/3, 160 00 Praha 6, Czech Republic, jan.mikula@cvut.cz, miroslav.kulich@cvut.cz.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

attracted towards *uncovered* areas until every area is *covered*. However, these solutions are of poor quality too (some can be improved by more than 20%, as we show later).

This paper proposes a novel heuristic solution framework for the d-WRP that, in contrast to standard *decoupling*, *covers* the environment by a set of carefully constructed overlapping continuous regions (polygons) instead of just points. These regions are an analogy to the points in the *decoupling* approach because a route that visits them all *covers* the whole environment. The problem with *decoupling* is that during the route optimization, the set of possible output solutions is already highly restricted; thus, it is unlikely to contain a solution close to optimum. In addition, there is no obvious way to determine a *promising* set of visit locations. Nothing of this is true for our framework. The space of possible solutions after assuring full *visual coverage* is much less restricted than in the case of *decoupling*. The reason is that solution is determined not only by the order of the regions but also by *how* each of these regions is visited. Since we are in a continuous domain, there is an infinite number of ways to visit each region in general. Empirically we have found out that the *most promising* regions are the largest ($\Rightarrow$ more ways to visit each region) and most overlapping ($\Rightarrow$ more ways to visit several regions at once) as possible. Also, the number of these regions should not be very large because the subsequent route optimization would be too computationally expensive. All of these criteria are considered by the region generating procedure we designed.

The subsequent problem of finding a minimum route that visits each polygonal region from a given set is known as the *traveling salesman problem with continuous neighborhoods* (TSPN) [8]. Since the literature does not yet provide any solver for the TSPN variant, where obstacles must be considered, and the neighborhoods can intersect, we had to design one such solver. Thus, a novel heuristic TSPN solver is our second major contribution. First, it solves a discretized version of the problem where the regions' borders are sampled by points, which is known as the *exactly-one-in-a-set generalized traveling salesman problem* (GTSP) [9]. Then, the GTSP solution is further improved back in the continuous domain by a modified *rubberband algorithm* [10]. Although the TSPN solution involves a phase where the problem is discretized, the framework *as a whole* makes the next step towards solving the d-WRP *continuously*.

The proposed framework is thoroughly experimentally evaluated on 24 instances of the d-WRP, compared to two reference algorithms from the literature, and shown to provide the highest-quality solutions. In addition, the framework can explicitly consider a non-point robot, which makes the problem more complex because, in general, the area that the robot sees is no longer the same as the area that is reachable by the robot. The capability to deal with such situations is demonstrated on several selected instances of the rd-WRP.

The rest of this paper is organized as follows. Previous related works are reviewed in Sec. II. The novel solution approach is proposed in Sec. III, and experimentally evaluated in Sec. IV. The last Sec. V is devoted to concluding remarks.

II. RELATED LITERATURE REVIEW

The *classic WRP* was firstly studied by Chin and Ntafos [1]. The authors attempt to prove that the problem in a general polygonal domain is NP-hard by reducing the *Euclidean TSP* to it. However, their proof does not stand as was later shown by Dumitrescu and Tóth [3] who provide the correct proof based on *rectilinear TSP* instead. Variants of the *classic WRP*, which restrict polygonal domain $\mathcal{W}$ somehow, have been studied extensively in the literature over the last three decades. In simple polygons, the problem is polynomially solvable; several exact [11], [12] and approximation [13], [14] algorithms have been proposed for that particular case and also for an even more special case where the simple polygon is rectilinear [1], [15]. More recently, a variant where the domain is the union of either lines or segments has been considered [16], [17]. So far the only polynomial-time approximation algorithm for the *classic WRP* in a polygon with holes was introduced by Mitchel [18]; it has an approximation factor of $\mathcal{O}(\log^2 n)$.

Packer [4] introduced a heuristic algorithm for computing watchman routes in a polygon with holes. Besides the single-agent case, he considered the case of k watchmen, i.e., the *multiple WRP* (MWRP), where either the total length traveled (MinSum), or the longest route (MinMax) is minimized. Packer's *decoupling* algorithm starts with solving the AGP where static guards with unlimited visibility range are placed into the polygonal domain for full *visual coverage*. Then, a minimum spanning tree created from the pairwise shortest paths between the guards is split to construct the initial k routes that are further improved by vertex substitutions to obtain the final solution.

In the *classic WRP*, a route from which boundary $\partial\mathcal{W}$ of environment $\mathcal{W}$ is visible is also the route from which every point in $\mathcal{W}$ is visible. Under limited visibility, this no longer holds. Because of this fact, Ntafos [2] distinguished two problems: 1) the *d-watchman route problem*, if one wants to see only boundary $\partial\mathcal{W}$ of the environment, and 2) the *d-sweeper route problem*, if one wants to see both $\partial\mathcal{W}$ and the interior of $\mathcal{W}$. Danner and Kavraki [19] heuristically solved the first of the two problems while assuming an additional visibility constraint on the maximum angle of incidence. Their *decoupling* algorithm firstly determines static guards that see the whole boundary and satisfy the visibility constraints, which is done in a randomized fashion by a modified *dual sampling* algorithm proposed in [20]. Then, the resulting route is obtained by solving the TSP on a pairwise shortest-paths graph whose vertices are the guards.

Faigl [6] deals with the *d-sweeper route problem*. Despite Ntafos's terminology, the author established for the problem an abbreviation d-WRP[1], that we adopted. Faigl itroduces a heuristic framework for the d-WRP that relies on the SOM [21]: a two-layered artificial neural network that provides a non-linear transformation of a high-dimensional input space into a lower dimensional discrete output space. Faigl's framework represents the route as a ring of connected weighted neurons that evolve in $\mathcal{W}$ while obstacles are avoided. An adaptation procedure based on SOM attracts the neurons

[1]According to Ntafos, the abbraviation should instead be d-SRP.

toward *uncovered* parts of $\mathcal{W}$ by considering the current *coverage* of $\mathcal{W}$ by the ring. The framework also tackles the problem's multi-agent MinMax-criterion version. The SOM-based approach is further improved in [7] with new adaptation rules and novel initialization of the neurons' weights that lead to faster convergence of the network.

This paper extends the definition of d-WRP by considering a circular robot of radius r, which yields the rd-WRP. Our extended definition is related to the *coverage path planning* (CPP), which is the task of determining a route that passes over all points of the environment while avoiding obstacles (see survey [22]). The CPP usually assumes that the robot's body is the same as the area that is being *covered* (sensed) by the robot, which corresponds to assuming $r = d$ in our definition. Although our framework can solve instances of the CPP to some degree (without guarantees of any kind), we do not aim to compete with algorithms specialized to the problem. Instead, we assume that the robot is smaller than the area it *covers visually*, i.e., $r < d$. In these cases, using the coverage planning algorithms would be problematic because they do not consider the notion of d-visibility.

Finally, recent work [23] solves a discrete version of the WRP on a grid with heuristic search. However, this work is not directly relevant to our problem because the environment is modeled by a discrete binary grid map, the robot is one-cell-sized, and a discrete line-of-sight function models its field of view. By contrast, we assume a polygonal, continuous environment, and a circular robot with radius r whose field of view is defined by d-*visibility*.

III. HEURISTIC FRAMEWORK FOR THE rd-WRP

In this section, we describe the proposed framework in detail. For readers' convenience, we explain many concepts using a simple example of an $r = 0.6\,\text{m}$ robot operating inside $8\,\text{m} \times 8\,\text{m}$ environment $\mathcal{W}$, shown in Fig. 2a ($\mathcal{W}$ is white, obstacles are black, the robot is navy blue). The proposed heuristic framework can be summarized as follows:

1) assuming a point robot, generate a *polygonal coverage* of $\mathcal{W}$, i.e., set of polygons $\boldsymbol{\mathcal{R}} = \{\mathcal{R} \,|\, \mathcal{R}$ is convex and fits in a circle of diameter $d\}$ such that

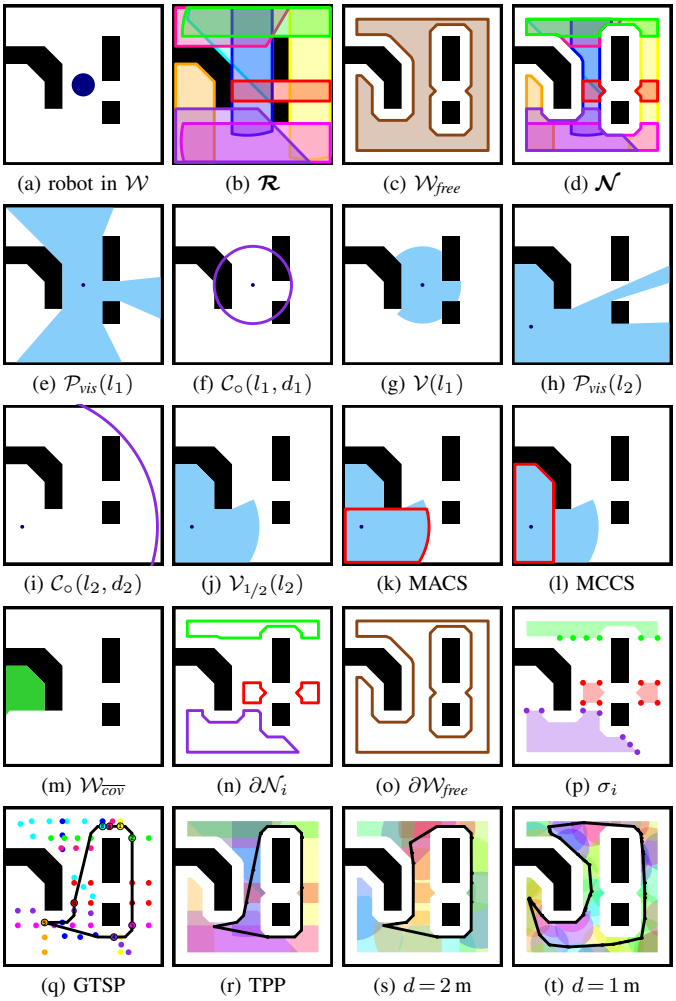
$$\bigcup_{\mathcal{R} \in \boldsymbol{\mathcal{R}}} \mathcal{R} = \mathcal{W}, \tag{1}$$

2) to consider a non-point robot, compute $\mathcal{W}_{free}$ by offsetting $\mathcal{W}$ by amount $-r$, otherwise set $\mathcal{W}_{free} = \mathcal{W}$, finally
3) get solution τ by solving the TSPN [8] with

$$\boldsymbol{\mathcal{N}} = \{\mathcal{R} \cap \mathcal{W}_{free} \,|\, \mathcal{R} \in \boldsymbol{\mathcal{R}}\} \setminus \{\varnothing\} \tag{2}$$

as the neighborhoods (note: $\boldsymbol{\mathcal{N}} = \boldsymbol{\mathcal{R}}$ for a point robot).

See the examples of $\boldsymbol{\mathcal{R}}$, $\mathcal{W}_{free}$, and $\boldsymbol{\mathcal{N}}$ in Fig. 2b-2d. Note that $\mathcal{W}_{free}$ represents both free workspace and free configuration space of the robot (recall that the robot is assumed holonomic and circular). The procedure for generating polygons $\boldsymbol{\mathcal{R}}$ is proposed in Subsec. III-A. The heuristic algorithm for solving the TSPN is proposed in Subsec. III-B.



(a) robot in $\mathcal{W}$ (b) $\boldsymbol{\mathcal{R}}$ (c) $\mathcal{W}_{free}$ (d) $\boldsymbol{\mathcal{N}}$
(e) $\mathcal{P}_{vis}(l_1)$ (f) $\mathcal{C}_\circ(l_1, d_1)$ (g) $\mathcal{V}(l_1)$ (h) $\mathcal{P}_{vis}(l_2)$
(i) $\mathcal{C}_\circ(l_2, d_2)$ (j) $\mathcal{V}_{1/2}(l_2)$ (k) MACS (l) MCCS
(m) $\mathcal{W}_{\overline{cov}}$ (n) $\partial\mathcal{N}_i$ (o) $\partial\mathcal{W}_{free}$ (p) σ_i
(q) GTSP (r) TPP (s) $d = 2\,\text{m}$ (t) $d = 1\,\text{m}$

Fig. 2: Examples of the most crucial structures and concepts.

A. Generating the polygonal coverage

The goal of the *polygonal coverage* is to create such set of polygons $\boldsymbol{\mathcal{R}}$ that visiting them all (by a point robot) would imply that the whole $\mathcal{W}$ has been seen. This requirement implies Eq. (1) and one more condition: (c_1) for all $\mathcal{R} \in \boldsymbol{\mathcal{R}}$ every pair of points $p_1, p_2 \in \mathcal{R}$ must be d-visible to each other. Apparently, all convex polygons that are fully inside $\mathcal{W}$ and fit inside a circle of diameter d satisfy condition (c_1). Therefore, e.g., a triangular mesh generated by *conforming constrained Delaunay triangulation* (CCDT) [24] could be used. However, the CCDT mesh would severely underestimate what part of the environment is seen by the robot when it visits a particular triangle. Also, if the environment's border is very complex, many fragment-like triangles appear in the mesh. Both of these properties may negatively influence the solution's quality and computational complexity of derived subproblems. Instead, we aim for a procedure that *covers* the environment with polygons satisfying condition (c_1), and in addition, it tries to minimize their number, maximize their size, and allows that they can intersect.

The *polygonal coverage* algorithm shown in Alg. 1 starts with empty set $\boldsymbol{\mathcal{R}}$ and uncovered region $\mathcal{W}_{\overline{cov}}$ initialized as the whole environment $\mathcal{W}$ (line 1). Then, inside the main

loop, it selects the best polygon $\mathcal{R}^\star$ (lines 3-11) to be added to $\boldsymbol{\mathcal{R}}$ (line 12) and clipped away from $\mathcal{W}_{\overline{cov}}$ (line 13). The process iterates until the ratio of the uncovered area is lesser than parameter $\epsilon_{\overline{cov}}$, which guarantees coverage of at least $100 \cdot (1 - \epsilon_{\overline{cov}})\%$. The selection of $\mathcal{R}^\star$ follows the *dual sampling* scheme [20]. The first seed l_0 is sampled randomly from $\mathcal{W}_{\overline{cov}}$ and the first candidate $\mathcal{R}_0$ is constructed from it. Then, i_{max} other candidates are constructed similarly by sampling only inside $\mathcal{R}_0$[2]. Finally, the best candidate $\mathcal{R}^\star$ is the one maximizing intersection with $\mathcal{W}_{\overline{cov}}$. When tie-breaking, the candidate with a larger area is selected.

Next, we detail how candidate polygons $\mathcal{R}_i$, are constructed on lines 4 and 8 of Alg. 1. The construction relies upon the concept of *visibility polygons*. *Visibility polygon* $\mathcal{P}_{vis}(l)$ at point $l \in \mathcal{W}$ is a set of all points of $\mathcal{W}$ that are *visible* from l assuming unrestricted visibility range (see the examples in Fig. 2e, 2h). We further consider a region sensed from l, denoted as $\mathcal{V}(l)$, defined as the union of $\mathcal{P}_{vis}(l)$ and circle[3] $\mathcal{C}_\circ(l, d)$ of radius d centered in l. See the construction of $\mathcal{V}$ in Fig. 2e-2g. The *dual sampling* algorithm utilizes region $\mathcal{V}_{1/2}$ defined as

$$\mathcal{V}_{1/2}(l) = \mathcal{C}_\circ(l, d/2) \cap \mathcal{P}_{vis}(l) \tag{3}$$

for given sample point l. See the construction of $\mathcal{V}_{1/2}$ in Fig. 2h-2j. Note, that $\mathcal{V}_{1/2}$ is constructed the same way as $\mathcal{V}$, except $\mathcal{P}_{vis}$ is clipped by a circle with half the radius. Region $\mathcal{V}_{1/2}$ is defined such that it fits inside a circle of diameter d to partially satisfy condition (c_1). However, it may be concave, so we can not use it as $\mathcal{R}_i$ in Alg. 1 directly. Instead, $\mathcal{R}_i$ is created by cutting off parts of $\mathcal{V}_{1/2}$, until the result is a convex polygon. The question is how to select proper cuts. One strategy shown in Fig. 2k is to aim for the maximal area of the result. This is equivalent to finding the *maximum area convex subset* (MACS) of $\mathcal{V}_{1/2}$.

Coeurjolly and Chasserythe [25] introduce an approximate algorithm for the MACS problem in *star-shaped* polygons. A polygon is *star-shaped* if it contains point l from which all other points of the polygon are *visible*, which is a property satisfied by $\mathcal{V}_{1/2}(l)$ from the definition. The algorithm [25] starts with a given *star-shaped* concave polygon. Then, according to several criteria, it determines a set of promising candidate cuts that would bring the polygon closer to convexity. Finally, from the candidate cuts, one that results in a maximal-area subset of the original polygon is selected and executed. This process repeats until a convex polygon is received. We adopt this procedure (using the same types of candidate cuts as [25]) to transform $\mathcal{V}_{1/2}$ into a convex polygon. However, in our case, we prefer cuts that maximize intersection with uncovered region $\mathcal{W}_{\overline{cov}}$, which is maintained by Alg. 1. Maximizing the area of the result is just a secondary rule (i.e., meant for tie-breaking) for selecting the best cut. We call the result *maximally covering convex subset* (MCCS) of $\mathcal{V}_{1/2}(l)$ *w.r.t.* $\mathcal{W}_{\overline{cov}}$.

[2]Experiments in [20] have shown that this *dual-sampling* approach finds better solutions for the AGP than plain *single-sampling*.

[3]Note that in principle, neither $\mathcal{C}_\circ$ nor $\mathcal{V}$ can be represented by polygons because polygons are composed of straight line segments, but circles or their parts are not. However, since our approach is heuristic, we can approximate circles by regular polygons (with many vertices) inscribed to them.

Fig. 2l displays an example of MCCS of $\mathcal{V}_{1/2}$ *w.r.t.* particular region $\mathcal{W}_{\overline{cov}}$, shown separately in Fig. 2m.

B. The heuristic algorithm for the TSPN

The last step of the proposed WRP framework is to find the shortest *collision-free* route visiting every region in set $\boldsymbol{\mathcal{N}}$ (Eq. (2)). This is equivalent to solving the TSPN [8] with $\boldsymbol{\mathcal{N}}$ as the neighborhoods. We propose a heuristic procedure shown in Alg. 2 to solve the TSPN while considering obstacles. The following notation is used: $\tau(s, g)$ is the shortest *collision-free* route connecting two points $s, g \in \mathcal{W}_{free}$, and $Len(\tau)$ is the length of route τ. The procedure consists of three basic steps: 1) sampling neighborhoods $\boldsymbol{\mathcal{N}}$ (lines 1-4), 2) determining order ς of the neighborhoods by solving the GTSP [9] (lines 5-6), and 3) obtaining final WRP solution τ by solving the *touring polygons problem* (TPP) [26] for the neighborhoods ordered according to ς (line 7). The following paragraphs further detail these steps.

First, we explain how the neighborhood sampling is done. Obviously, when visiting $\mathcal{N}$, the visitor must first cross (or touch) its border $\partial\mathcal{N}$. Therefore, it is enough to sample just $\partial\mathcal{N}$. Furthermore, $\mathcal{N}$ could share some parts of its border with $\mathcal{W}_{free}$. However, a robot with radius r can not enter $\mathcal{N}$ by crossing $\partial\mathcal{W}_{free}$ because that would imply that its previous configuration was outside its free configuration space. Thus, $\partial\mathcal{N} \cap \partial\mathcal{W}_{free}$ can be excluded from sampling. The strategy for sampling the remaining $\partial\mathcal{N} \setminus \partial\mathcal{W}_{free}$ is to decompose it into maximal connected components, which can be represented as polylines, and sample them individually. If $\partial\mathcal{N} \setminus \partial\mathcal{W}_{free}$ is composed of only one such polyline that is closed and whose length is smaller than parameter d_{sample}, then $\sigma_\mathcal{N} = \langle$centroid of $\mathcal{N}\rangle$. Otherwise, the polylines are sampled equidistantly such that the distance (along the polyline) between two samples is maximal but no bigger than d_{sample}. For open polylines, the first and last vertices are always included. An example of sampling three neighborhoods is shown in Fig. 2n-2p.

Algorithm 1: *Polygonal coverage* of $\mathcal{W}$.

Input: $\mathcal{W}$ (polygonal enviroment), d (visibility radius)
Output: $\boldsymbol{\mathcal{R}}$ (polygons covering $\mathcal{W}$)
Param: i_{max} (# of dual sampling iterations), $\epsilon_{\overline{cov}}$ (goal coverage)

1 Initialize $\boldsymbol{\mathcal{R}} \leftarrow \varnothing$, and $\mathcal{W}_{\overline{cov}} \leftarrow \mathcal{W}$.
2 **while** $Area(\mathcal{W}_{\overline{cov}})/Area(\mathcal{W}) > \epsilon_{\overline{cov}}$ **do**
3 Select point $l_0 \in \mathcal{W}_{\overline{cov}}$ randomly.
4 $\mathcal{R}_0 \leftarrow$ MCCS of $\mathcal{V}_{1/2}(l_0)$ *w.r.t.* $\mathcal{W}_{\overline{cov}}$
5 $\mathcal{R}^\star,\ c^\star,\ a^\star \leftarrow \mathcal{R}_0,\ Area(\mathcal{R}_0 \cap \mathcal{W}_{\overline{cov}}),\ Area(\mathcal{R}_0)$
6 **for** $i \leftarrow 1, \ldots, i_{max}$ **do**
7 Select point $l_i \in \mathcal{R}_0$ randomly.
8 $\mathcal{R}_i \leftarrow$ MCCS of $\mathcal{V}_{1/2}(l_i)$ *w.r.t.* $\mathcal{W}_{\overline{cov}}$
9 $c_i,\ a_i \leftarrow Area(\mathcal{R}_i \cap \mathcal{W}_{\overline{cov}}),\ Area(\mathcal{R}_i)$
10 **if** $c_i > c^\star$ *or* $(c_i = c^\star$ *and* $a_i > a^\star)$ **then**
11 $\mathcal{R}^\star,\ c^\star,\ a^\star \leftarrow \mathcal{R}_i,\ c_i,\ a_i$
12 $\boldsymbol{\mathcal{R}} \leftarrow \boldsymbol{\mathcal{R}} \cup \{\mathcal{R}^\star\}$
13 $\mathcal{W}_{\overline{cov}} \leftarrow \mathcal{W}_{\overline{cov}} \setminus \mathcal{R}^\star$

By sampling all neighborhoods, one obtains an instance of the GTSP, where the sample points represent vertices of a graph and affiliations to neighborhoods determine how they are grouped. It is reasonable to assume that the graph is complete if we also assume that $\mathcal{W}_{free}$ has exactly one maximal connected component. Edges of the graph are weighted according to the lengths of the shortest paths in $\mathcal{W}_{free}$ between the vertices. A solution of the GTSP is a circuit (i.e., a closed walk) in the graph that visits exactly one vertex from each group (except for the first and last, which are the same) and minimizes the total weight along the edges. Recall that the classic TSP can be defined by single distance matrix $D=(d_{ij})$; similarly, the GTSP can be defined by D and mapping ν between the vertices and groups to which they belong (referring to lines 4-6 of Alg. 2). For solving the GTSP, we use GLNS: an effective *large neighborhood search* heuristic introduced by Smith and Imeson [27]. An example of a GTSP instance (colored points) and a GTSP solution (black tour) is shown in Fig. 2q.

The last step of the TSPN algorithm is to refine the solution route by solving the TPP. The TPP is the problem of finding point $p_i \in \mathcal{N}_i$ in each neighborhood $\mathcal{N}_i \in \boldsymbol{\mathcal{N}}$ such that closed *collision-free* route τ visiting all such points in an order prescribed by $\varsigma(\boldsymbol{\mathcal{N}}) = \langle \mathcal{N}_1, \ldots, \mathcal{N}_n \rangle$ is the shortest possible. We use a modified version of the algorithm introduced in Pan et al. [10]. The authors deal with the TPP while assuming that the polygons are pairwise disjoint and no obstacles are present. We apply a modified version of their *rubberband algorithm* to our variant of the TPP with intersecting polygons and the necessity to avoid obstacles. The algorithm is shown in Alg. 3. The input, i.e., ordered sequence of neighborhoods $\varsigma(\boldsymbol{\mathcal{N}})$, is obtained by looking at the group labels of the GTSP solution received at line 6 of Alg. 2. Similarly, the initial sequence of points π_0 is taken from the same GTSP solution as the points that were selected from each group. Notation $\tau(\pi)$ denotes a closed route that connects all consecutive points in sequence $\pi = \langle p_1, \ldots, p_n \rangle$ and, in addition, p_n and p_1, by the shortest *collision-free* sub-routes. The algorithm iteratively modifies point sequence π such that $\tau(\pi)$ is getting shorter. In each iteration k, the algorithm starts with a sequence from previous iteration $k-1$. Then, it goes over all triplets $i=0,\ldots,n-1$ of consecutive points (denoted as s, p_i, and g) in the sequence and computes a new value for the middle point (p_i). The new value is a point that lies on one of the polylines defined as $\partial\mathcal{N}_{i+1} \setminus \partial\mathcal{W}_{free}$ and minimizes the total cost of travel (route's length) from s to itself and then to g. An example of a TPP solution is shown in Fig. 2r. The TPP heuristic, in general, improves[4] the WRP solution obtained initially by solving the GTSP. Finally, see different solutions computed by the framework while assuming different d's in Fig. 2s-2t.

Algorithm 2: Heuristic procedure for the TSPN.

Input: $\boldsymbol{\mathcal{N}}$ (polygonal neighborhoods), $\mathcal{W}_{free}$ (free config. space)
Output: τ (solution path)
Param: d_{sample} (maximal sampling distance)

1 Initialize empty seq. of samples σ, and neighborhoods ν.
foreach $\mathcal{N} \in \boldsymbol{\mathcal{N}}$ **do**
2 Sample $\partial\mathcal{N} \setminus \partial\mathcal{W}_{free}$ using parameter d_{sample} and save the samples to $\sigma_{\mathcal{N}}$.
3 **foreach** $s \in \sigma_{\mathcal{N}}$ **do**
4 Append s, $\mathcal{N}$ to the end of σ, ν, respectively.
5 Compute distance matrix $D = (d_{ij} = \mathit{Len}(\tau(s_i, s_j)) \,|\, \forall s_i, s_j \in \sigma)$.
6 Get sequence of neighborhoods $\varsigma(\boldsymbol{\mathcal{N}})$ by solving the GTSP defined by D and ν.
7 Get WRP solution τ by solving the TPP for $\varsigma(\boldsymbol{\mathcal{N}})$.

Algorithm 3: *Rubberband algorithm* for the TPP.

Input: $\varsigma(\boldsymbol{\mathcal{N}}) = \langle \mathcal{N}_1, \ldots, \mathcal{N}_n \rangle$ (ordered neighborhoods), $\pi_0 = \langle p_i \,|\, p_i \in \mathcal{N}_i \rangle$ (point sequence), $\mathcal{W}_{free}$
Output: τ (solution path)
Param: k_{max} (# of improvement attempts)

1 Initialize $\pi^\star \leftarrow \pi_0$.
2 **for** $k \leftarrow 1, \ldots, k_{max}$ **do**
3 Let $\pi_{k-1} \equiv \langle p_0, \ldots, p_{n-1} \rangle$.
4 **for** $i \leftarrow 0, \ldots, n-1$ **do**
5 $s,\ g \leftarrow p_{(i-1 \bmod n)},\ p_{(i+1 \bmod n)}$
6 $p_i \leftarrow \underset{q \in \partial\mathcal{N}_{i+1} \setminus \partial\mathcal{W}_{free}}{\arg\min} \big(\mathit{Len}(\tau(s,q)) + \mathit{Len}(\tau(q,g))\big)$
7 $\pi_k \leftarrow \langle p_0, \ldots, p_{n-1} \rangle$
8 **if** $\mathit{Len}(\tau(\pi_k)) < \mathit{Len}(\tau(\pi^\star))$ **then**
9 $\pi^\star \leftarrow \pi_k$
10 $\tau \leftarrow \tau(\pi^\star)$

IV. COMPUTATIONAL EVALUATION

To the best of our knowledge, the improved SOM-based method presented in [7] is the latest published solution to the d-WRP in a polygon with holes. Therefore, it makes a valid reference for comparison with our novel approach. Furthermore, in the same work, the authors compare their SOM to a simple *decoupling* algorithm. The locations are generated by the deterministic *sensor-placement algorithm* [28] and the TSP on the pairwise shortest-paths graph is solved by the exact solver Concorde [29]. However, better decoupling approaches exist [19]. For example, the *dual sampling* algorithm [20] can generate the locations, and the LKH heuristic [30] can solve the TSP[5]. The latter description yields our second reference algorithm called DS+LKH.

The propsed heuristic framework for the rd-WRP was implemented in C++ using the C++17 standard. The framework computes *visibility polygons* using our implementation of the *triangular expansion* algorithm [31], which uses a *Delaunay triangulation* mesh generated by *Triangle* [32]. *Clipper* [33] is employed for computing polygon operations

[4]Note that the improvement in Fig. 3 is slight (the TPP solution is just about 0.2% shorter) because of the environment's simplicity. However, the obtained improvements are much more significant (4.2% ± 3.8%) for larger and more complex environments tested in Sec. IV (Tab. I).

[5]The *dual sampling* algorithm [20] generates better guards than the *sensor-placement algorithm* [28], and the LKH heuristic [30] improves the speed over Concorde [29] without significantly reducing the solution quality.

TABLE I: Experimental results for the d-WRP. Colors denote comparison of the algorithms: 1st (best), 2nd, 3rd, 4th (worst).

Map	d [m]	l_{ref} [m]	SOM [7]			DS+LKH [19]				Proposed (*Trade-off*)					Proposed (*Best*)				
			PDM	PDB	$\overline{t}$ [βs]	$\overline{n}$	PDM	PDB	$\overline{t}$ [s]	$\overline{n}$	$\overline{r}_\sigma$	PDM	PDB	$\overline{t}$ [s]	$\overline{n}$	$\overline{r}_\sigma$	PDM	PDB	$\overline{t}$ [s]
jh	∞	193.3	-45.3	-49.6	0.1	28	-36.1	-42.3	0.2	59	6.5	-51.7	-53.0	2.0	56	14.7	-51.6	-53.0	19.0
	10.0	194.6	-47.0	-49.6	0.1	30	-35.0	-40.9	0.2	62	5.6	-51.6	-52.8	1.7	60	12.6	-51.4	-53.3	15.6
	5.0	204.3	-43.2	-48.3	0.1	35	-29.5	-35.5	0.2	93	5.0	-50.5	-51.5	1.8	84	10.7	-51.5	-52.2	26.8
	4.0	207.9	-33.7	-38.6	0.2	40	-28.1	-33.7	0.2	123	4.3	-46.7	-49.4	2.0	117	10.0	-49.0	-49.6	61.3
	3.0	215.5	-17.8	-21.9	0.5	53	-23.2	-29.6	0.2	191	3.4	-33.1	-36.7	2.6	173	8.7	-42.4	-43.2	114.2
	2.0	295.5	-7.5	-12.2	2.6	101	-20.7	-23.5	0.4	353	2.3	-24.1	-26.1	4.8	343	6.5	-33.0	-34.5	135.0
	1.5	359.0	-1.8	-5.4	7.1	152	-21.7	-23.9	0.8	587	2.0	-19.0	-22.9	10.7	566	5.2	-26.5	-29.2	169.9
	1.0	458.0	2.3	-0.7	45.4	334	-14.6	-15.8	2.6	1198	1.6	-7.4	-11.7	44.2	1163	4.0	-19.2	-21.8	358.9
ta	∞	215.6	-33.3	-34.6	0.0	10	-29.9	-34.4	0.0	32	7.2	-40.5	-41.0	1.2	30	22.3	-40.3	-41.0	4.3
	10.0	216.9	-32.0	-33.0	0.0	14	-25.1	-33.5	0.0	52	5.4	-35.2	-36.9	1.2	50	17.5	-34.9	-37.4	9.3
	5.0	256.8	-14.9	-18.0	0.1	32	-16.2	-21.4	0.1	110	3.4	-24.1	-25.6	1.5	105	12.4	-27.9	-29.3	49.0
	4.0	291.3	-9.8	-13.8	0.2	45	-17.1	-21.5	0.1	156	2.9	-23.8	-26.7	1.7	150	10.8	-28.1	-29.9	110.6
	3.0	335.6	-4.8	-9.1	0.5	73	-16.2	-19.9	0.3	247	2.1	-19.9	-25.1	2.5	239	8.9	-26.9	-28.5	124.3
	2.0	427.0	0.3	-3.3	3.9	142	-19.7	-22.3	0.7	502	1.7	-15.5	-19.2	6.5	489	6.6	-26.8	-28.7	181.1
	1.5	538.5	-1.6	-4.4	19.2	240	-22.8	-24.6	1.5	854	1.5	-14.1	-18.5	19.0	833	5.0	-26.6	-28.2	282.3
	1.0	774.3	-0.1	-1.5	138.0	495	-27.8	-28.6	5.1	1823	1.3	-15.5	-19.6	104.2	1776	3.9	-26.0	-30.1	971.3
pb	∞	554.9	-20.5	-22.5	0.2	13	-31.7	-35.8	0.0	32	7.3	-38.7	-38.7	1.2	31	26.0	-38.7	-38.8	4.2
	10.0	615.9	-14.6	-16.7	0.2	37	-8.1	-11.3	0.1	96	4.5	-17.8	-18.8	1.4	94	12.0	-19.6	-20.0	28.7
	5.0	687.2	-7.7	-9.3	0.5	83	-4.8	-7.8	0.2	212	3.7	-11.7	-12.5	2.2	207	10.0	-12.8	-13.4	113.3
	4.0	721.9	-6.8	-7.9	1.0	106	-6.0	-7.8	0.3	300	3.5	-10.4	-12.1	3.2	284	9.5	-12.6	-13.3	127.2
	3.0	781.6	-6.3	-7.0	4.1	153	-7.5	-8.4	0.5	492	2.9	-9.1	-10.3	7.0	456	8.4	-13.7	-14.3	176.9
	2.0	919.0	-5.4	-6.5	22.3	283	-13.2	-14.9	2.1	1089	1.8	-3.5	-7.7	29.6	1064	6.3	-16.9	-18.2	475.0
	1.5	1158.2	-3.2	-4.1	112.2	469	-21.1	-22.3	5.3	1783	1.7	-9.6	-14.1	89.4	1720	5.0	-24.8	-25.9	940.2
	1.0	1606.6	-1.4	-2.0	889.3	1144	-17.3	-18.0	24.6	3835	1.4	-13.8	-16.6	561.5	3736	3.9	-23.6	-25.8	4425.1

PDM = PD(l_{mean}), PDB = PD(l_{best}), PD(l) := $100 \cdot (l - l_{ref})/l_{ref}$, $\overline{t}$—mean time, $\overline{n}$—mean # of generated locations/polygons, $\overline{r}_\sigma$—mean # of samples per neighborhood.

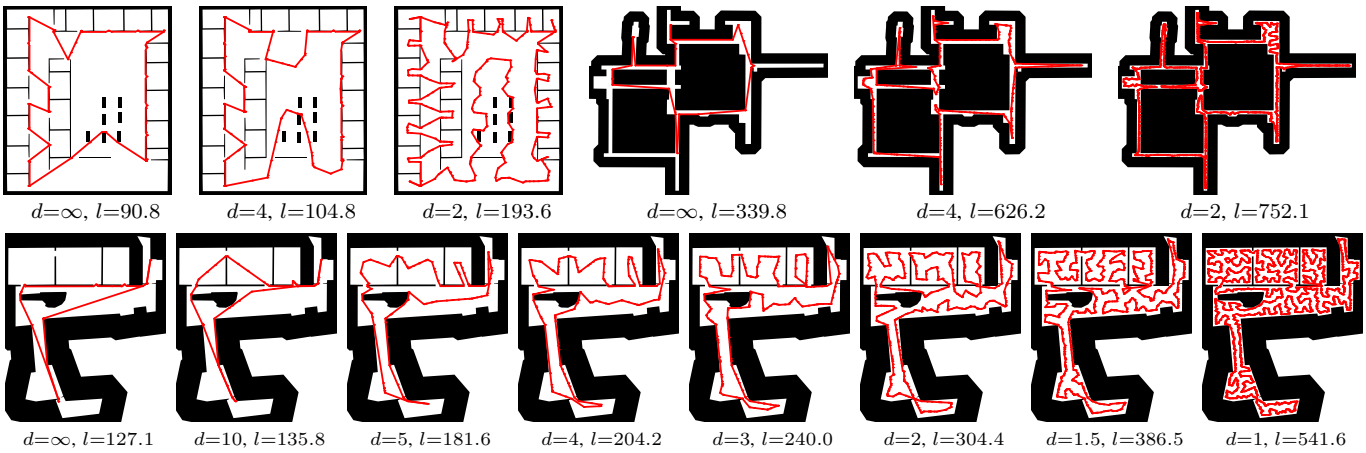


Fig. 3: Selected best d-WRP solutions found by the proposed framework for maps: *jh* (top-left), *pb* (top-right), *ta* (bottom).

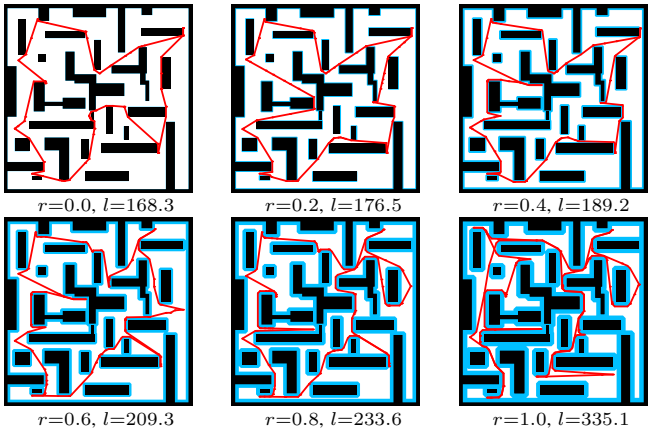


Fig. 4: rd-WRP solutions in a $40\,\text{m} \times 40\,\text{m}$ map for $d = \infty$ and increasing r. $\mathcal{W}_{free}$, $\mathcal{W}_{obs} = \mathcal{W} \setminus \mathcal{W}_{free}$ are white, blue, respectively. The *visual coverage of* $\mathcal{W}$ is 100 % in all cases.

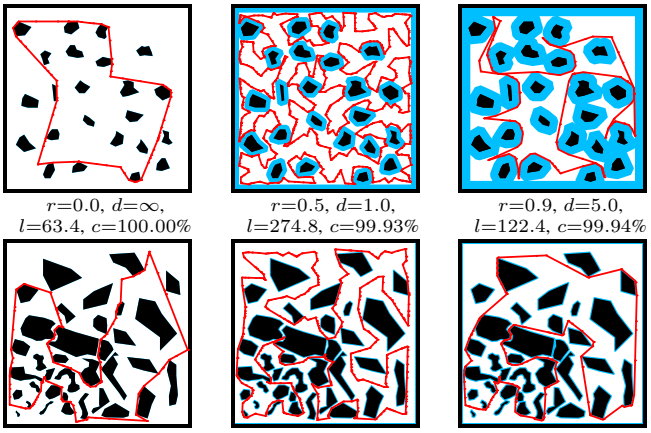


Fig. 5: Selected rd-WRP solutions in less structured environments (c is the percentage *visual coverage of* $\mathcal{W}$).

and offsetting. A standard combination of a visibility graph computed by the algorithm proposed in [34] and Dijkstra's algorithm is used for obtaining distance matrix D (recall line 5 of Alg. 2). The authors' Julia implementation of GLNS is rewritten to C++ while preserving the structure and parameters of the program.

We consider two parametrizations of our framework:

- *Trade-off*: i_{max}=10, d_{sample}=10 m, k_{max}=10, t_{GLNS}=1 s, which provides a trade-off between solution quality and runtime, and
- *Best*: i_{max}=100, d_{sample}=1 m, k_{max}=100, t_{GLNS}=100 s, which aims at finding the best quality solutions regardless of runtime.

The first three parameters are from Alg. 1-3, and t_{GLNS} is an upper runtime limit posed on GLNS (GLNS is an anytime algorithm). Parameter value $i_{max} = 100$ is also used in the reference algorithm DS+LKH. In addition, all parametrizations share $\epsilon_{\overline{cov}} = 0.001$ (Alg. 1), the *default* mode of GLNS (GLNS has three modes: *slow*, *default*, and *fast*; see [27]), and number of vertices $n_\circ = 32$ of the regular polygons used to approximate circles (e.g., in Eq. (3)).

The proposed, and two reference algorithms are evaluated and compared on several d-WRP instances. Moreover, we show at the end how the proposed algorithm can adapt to a robot with non-zero radius r and less structured environments. For the main experiments, we use the same polygonal environments: *jh* (20.6 m × 23.2 m), *ta* (39.6 m × 46.8 m), *pb* (133.3 m × 104.8 m), and visibility ranges $d \in \{\infty, 10.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0\}$ m as in [7]. All the algorithms are randomized; thus, 20 solutions are found for every instance-algorithm pair. The solutions' quality is measured using two metrics: $\text{PDM} = \text{PD}(l_{mean})$, and $\text{PDB} = \text{PD}(l_{best})$, where l_{mean}, and l_{best} are the mean and best solution lengths, respectively, and $\text{PD}(l) \coloneqq 100 \cdot (l - l_{ref})/l_{ref}$ is the percent deviation from reference path length l_{ref}[6]. Besides, $\overline{n}$ denotes the mean number of generated sensing locations for the DS+LKH. The same symbol also denotes the mean number of polygons generated by Alg. 1. Furthermore, $\overline{r}_\sigma$ represents the mean number of samples per neighborhood generated in Alg. 2.

Most of the experiments were executed within the same computational environment using a single core of the Intel Core i7-6700 CPU (3.40 GHz), 16 GB of RAM, and running Ubuntu 20.04. Experiments with the SOM method make the exception because the authors' implementation was not made public. Thus, we show the runtimes presented in [7] corrected by factor $\beta = \frac{914}{2302} \approx 0.397$ based on the single-thread CPU ratings obtained from *cpubenchmark.net*[7]. With this correction, the presented runtimes can be directly compared.

The results are shown in Tab. I. The first thing one can notice is that the methods' relative performance heavily depends on the considered visibility range d. Conclusions derived from experiments with either $d = \infty$ or $d = 1$ m would significantly differ. However, one observation is consistent regardless of d: the proposed (*Best*) provides the best quality solutions at the cost of being the slowest. On average, it provides (10.6 ± 6.6)%, (16.2 ± 8.6)%, and (23.2 ± 4.7)% better solutions than the SOM for $d \geq 10$, $5 \geq d \geq 3$, and $d \leq 2$, respectively. Similarly, it provides (16.7 ± 6.4)%, (16.4 ± 9.5)%, and (6.0 ± 4.7)% better solutions than DS+LKH for the same values of d, respectively. The runtime of the proposed framework can be significantly reduced by considering *Trade-off* parametrization. This parametrization provides solutions within 2, and 7 seconds for $d \geq 10$, and $5 \geq d \geq 3$, respectively. Regarding quality, it provides (10.4 ± 6.9)%, (11.6 ± 6.5)%, and (11.8 ± 6.4)% better solutions than the SOM for $d \geq 10$, $5 \geq d \geq 3$, and $d \leq 2$, respectively. Compared to DS+LKH, it provides (16.6 ± 6.8)%, and (11.6 ± 9.8)%, better solutions for $d \geq 10$, and $5 \geq d \geq 3$, respectively. On the other hand, for $d \leq 2$, the solutions are worse by (7.9 ± 6.5)% than the solutions produced by DS+LKH. We believe that the observed improvement, i.e., more than 10% on average for $d \geq 3$, is meaningful, e.g., in robotics, because modern sensors are expected to have $d \geq 3$, and a reasonable frequency of the robot's mission-planning module (incorporating our framework) can be about 0.1 Hz. Selected best solutions found by the proposed framework are shown in Fig. 3.

TABLE II: Detailed view of the runtime.

Map	d	Proposed (*Trade-off*)					Proposed (*Best*)				
		$\overline{t}$ [s]	% of time				$\overline{t}$ [s]	% of time			
			cov	*dij*	*tspn*	*rest*		*cov*	*dij*	*tspn*	*rest*
jh	∞	2.0	28.4	7.7	58.7	5.2	19.0	25.1	14.2	60.1	0.6
	10.0	1.7	16.0	7.3	71.0	5.7	15.6	12.2	12.1	75.0	0.7
	5.0	1.8	14.9	10.7	67.5	6.9	26.8	4.5	10.5	84.5	0.5
	4.0	2.0	19.0	12.3	61.1	7.6	61.3	2.4	8.4	88.9	0.3
	3.0	2.6	28.8	13.3	49.8	8.1	114.2	1.7	8.2	89.9	0.2
	2.0	4.8	49.0	15.5	29.1	6.4	135.0	2.8	20.3	76.6	0.3
	1.5	10.7	57.1	24.8	14.2	3.9	169.9	4.7	33.6	61.4	0.3
	1.0	44.2	62.5	31.7	4.3	1.5	358.9	8.5	61.6	29.6	0.3
ta	∞	1.2	8.9	2.8	84.0	4.3	4.3	20.4	19.5	58.8	1.3
	10.0	1.2	6.5	3.7	84.1	5.7	9.3	5.2	14.0	80.0	0.8
	5.0	1.5	15.2	5.6	72.4	6.8	49.0	1.4	10.5	87.9	0.2
	4.0	1.7	23.3	7.4	62.4	6.9	110.6	0.9	8.7	90.3	0.1
	3.0	2.5	41.5	7.3	45.0	6.2	124.3	1.4	17.2	81.2	0.2
	2.0	6.5	65.8	12.4	18.3	3.5	181.1	3.1	40.7	56.1	0.1
	1.5	19.0	69.5	21.6	7.2	1.7	282.3	5.4	58.2	36.3	0.1
	1.0	104.2	68.5	29.3	1.7	0.5	971.3	7.9	81.0	11.0	0.1
pb	∞	1.2	6.7	2.2	85.1	6.0	4.2	14.9	13.9	69.3	1.9
	10.0	1.4	8.8	6.0	75.5	9.7	28.7	1.5	5.4	92.6	0.5
	5.0	2.2	25.8	15.3	49.9	9.0	113.3	0.9	10.0	88.9	0.2
	4.0	3.2	36.1	21.8	34.6	7.5	127.2	1.4	19.0	79.3	0.3
	3.0	7.0	48.7	30.0	16.7	4.6	176.9	2.4	40.1	57.3	0.2
	2.0	29.6	66.4	26.7	5.1	1.8	475.0	4.8	73.3	21.7	0.2
	1.5	89.4	64.1	32.2	2.9	0.8	940.2	6.4	82.1	11.3	0.2
	1.0	561.5	62.6	36.1	1.0	0.3	4425.1	8.4	88.1	3.4	0.1

A detailed view of the proposed framework runtime can be seen in Tab. II. The table displays a percentage share of runtime for the following parts of the algorithm: *cov* ∼ *polygonal coverage* of $\mathcal{W}$., *dij* ∼ Dijkstra's algorithm, *tspn* ∼ solving the GTSP and TPP, and *rest* ∼ the rest (e.g., computing $\mathcal{W}_{free}$, visibility graph, sampling). Note that Dijkstra's algorithm takes up to 88% of runtime with decreasing d, which makes it the main bottleneck that reduces the scalability of the framework. However, this issue is not caused by the algorithm itself but rather by the fact that

[6]The value of l_{ref} taken from [7] is computed by the deterministic *decoupling algorithm* (*sensor-placement* [28] + Concorde [29]).

[7]The full CPU comparison: *https://www.cpubenchmark.net/compare/AMD-Athlon-Dual-Core-5050e-vs-Intel-i7-6700*

the number of samples (hence nodes of the graph supplied to Dijkstra's) does not scale well with decreasing d (try computing $\overline{n} \cdot \overline{r}_\sigma$ in Tab. I). We want to address this in the future by more selective sampling and smarter adaptive parametrization of the framework. Alternatively, we could try to replace the current approach for computing the all-pairs shortest paths before GTSP by some approximation as in [7].

Assuming $r = 0$, the proposed framework guarantees at least $100 \cdot (1 - \epsilon_{\overline{cov}})\%$ *visual coverage* of $\mathcal{W}$. However, with non-zero r, it may happen that certain $\mathcal{R} \in \boldsymbol{\mathcal{R}}$ has no intersection with $\mathcal{W}_{free}$, breaking the *visual coverage* warranty. Nevertheless, in practice, the framework provides satisfactory results as shown in Fig. 4 ($\mathcal{W}$: $40\,\mathrm{m} \times 40\,\mathrm{m}$). Here, our framework still achieves nearly 100% *visual coverage* of the non-inflated environment (white and blue) and preserves certain features of high-quality solutions (e.g., peaking behind a corner of a non-inflated obstacle), despite the fact that the inflation highly limits the robot's mobility and even makes some corridors impassable. Other example solutions of the rd-WRP in less structured environments (top: $20\,\mathrm{m} \times 20\,\mathrm{m}$, bottom: $40\,\mathrm{m} \times 40\,\mathrm{m}$) for various values of d, and r are shown in Fig. 5.

V. CONCLUSIONS AND FUTURE WORK

We introduced a novel heuristic framework for the challenging WRP in polygon with holes $\mathcal{W}$. The framework guarantees $100 \cdot (1 - \epsilon_{\overline{cov}})\%$ *visual coverage* of $\mathcal{W}$, where $\epsilon_{\overline{cov}} \in [0, 1]$ is a user-defined value, while considering d-*visibility* constraint (d-WRP). Also, it easily considers an agent with non-zero radius r without guarantees but achieving high percentage *visual coverage* in practice. The framework was thoroughly experimentally evaluated on 24 instances of the d-WRP and compared to two other major approaches from the literature. In terms of solution quality, the proposed framework is superior to the previous methods. Our future work will address improving runtimes and scalability of the framework and making it fully continuous. We want to achieve the latter by modifying GLNS [27] to solve the TSPN continuously, replacing the current procedure that solves the combination of discrete GTSP and continuous TPP.

REFERENCES

[1] W. Chin and S. Ntafos, "Optimum watchman routes," in *Proceedings of the second annual symposium on Computational geometry*. New York, USA: ACM Press, 1986, pp. 24–33.

[2] S. Ntafos, "Watchman routes under limited visibility," *Computational Geometry: Theory and Applications*, vol. 1, no. 3, pp. 149–170, 1992.

[3] A. Dumitrescu and C. Tóth, "Watchman tours for polygons with holes," *Computational Geometry*, vol. 45, no. 7, pp. 326–333, 2012.

[4] E. Packer, "Computing Multiple Watchman Routes," in *Experimental Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5038 LNCS, pp. 114–128.

[5] E. Acar, H. Choset, and Ji Yeong Lee, "Sensor-based coverage with extended range detectors," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 189–198, 2006.

[6] J. Faigl, "Approximate solution of the multiple watchman routes problem with restricted visibility range," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1668–1679, 2010.

[7] J. Faigl and L. Přeučil, "Inspection planning in the polygonal domain by Self-Organizing Map," *Applied Soft Computing*, vol. 11, no. 8, pp. 5028–5041, 2011.

[8] E. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197–218, 1994.

[9] S. Srivastava, S. Kumar, R. Garg, and P. Sen, "Generalized traveling salesman problem through n sets of nodes," *CORS Journal*, vol. 7, pp. 97–101, 1969.

[10] X. Pan, F. Li, and R. Klette, "Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons," *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry*, pp. 175–178, 2010.

[11] W. Chin and S. Ntafos, "Shortest watchman routes in simple polygons," *Discrete & Computational Geom.*, vol. 6, no. 1, pp. 9–31, 1991.

[12] S. Carlsson, H. Jonsson, and B. Nilsson, "Finding the Shortest Watchman Route in a Simple Polygon," *Discrete & Computational Geometry*, vol. 22, no. 3, pp. 377–402, 1999.

[13] B. Nilsson, "Approximating a Shortest Watchman Route." *Fundam. Inform.*, vol. 45, pp. 253–281, 2001.

[14] X. Tan, "A linear-time 2-approximation algorithm for the watchman route problem for simple polygons," *Theoretical Computer Science*, vol. 384, no. 1, pp. 92–103, 2007.

[15] H. Hoorfar and A. Bagheri, "A Linear-time Algorithm for Orthogonal Watchman Route Problem with Minimum Bends," 2017.

[16] A. Dumitrescu, J. Mitchell, and P. Żyliński, "Watchman routes for lines and line segments," *Computational Geometry*, vol. 47, no. 4, pp. 527–538, 2014.

[17] X. Tan and B. Jiang, "An improved algorithm for computing a shortest watchman route for lines," *Information Processing Letters*, vol. 131, pp. 51–54, 2018.

[18] J. Mitchell, "Approximating watchman routes," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia: Society for Industrial and Applied Mathematics, 2013, pp. 844–855.

[19] T. Danner and L. Kavraki, "Randomized planning for short inspection paths," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2000, pp. 971–976.

[20] H. Gonzalez-Banos and J. Latombe, *Planning Robot Motions for Range-Image Acquisition and Automatic 3D Model Construction*. Citeseer, 1998.

[21] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[22] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[23] T. Yaffe, S. Skyler, and A. Felner, "Suboptimally Solving the Watchman Route Problem on a Grid with Heuristic Search," *Fourteenth International Symposium on Combinatorial Search*, vol. 12, no. 1, pp. 106–114, 2021.

[24] J. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry: Theory and Applications*, vol. 22, pp. 21–74, 2002.

[25] D. Coeurjolly and J. Chassery, "Fast approximation of the maximum area convex subset for star-shaped polygons," 2004.

[26] M. Dror, A. Efrat, A. Lubiw, and J. Mitchell, "Touring a sequence of polygons," in *Proceed. of the thirty-fifth ACM symposium on Theory of computing*. New York, USA: ACM Press, 2003, pp. 473–483.

[27] S. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem," *Computers and Operations Research*, vol. 87, pp. 1–19, 2017.

[28] G. Kazazakis and A. Argyros, "Fast positioning of limited-visibility guards for the inspection of 2D workspaces," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2843–2848.

[29] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Concorde TSP Solver." [Online]. Available: https://www.math.uwaterloo.ca/tsp/concorde.html

[30] K. Helsgaun, "An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems," Tech. Rep., 2017.

[31] F. Bungiu, M. Hemmer, J. Hershberger, K. Huang, and A. Kröller, "Efficient Computation of Visibility Polygons," 2014.

[32] J. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," in *Applied Computational Geometry: Towards Geometric Engineering*, ser. Lecture Notes in Computer Science, 1996, vol. 1148, pp. 203–222.

[33] A. Johnson, "Clipper—an open source freeware library for clipping and offsetting lines and polygons." [Online]. Available: http://www.angusj.com/delphi/clipper.php

[34] M. Overmars and E. Welzl, "New methods for computing visibility graphs," *Proceedings of the 4th Annual Symposium on Computational Geometry*, pp. 164–171, 1988.