Speeding Up Coverage Queries in 3D Multi-Goal Path Planning

Petr Janoušek, Jan Faigl

Abstract— **In this paper, we present a supporting structure for speeding up visibility queries needed for a 3D multi-goal path planning arising from a robotic coverage problem where goals are sensing locations from which an object of interest can be covered. Although such coverage problems can be addressed by a decomposed approach where sensing locations are determined prior finding the sequence of their visits, the proposed approach is motivated by a solution of the problem in which sensing locations are simultaneously determined together with evaluation of the path connecting them in order to provide a cost effective inspection path. The proposed structure divides the space into elements that support determination of suitable sensing locations to cover the objects during solution of the multi-goal path planning.**

I. Introduction

A wide range of practical robotic applications can be formulated as the multi-goal path planning problem, which stands to determine the cost effective path visiting a set of goal locations. Then, the robot is requested to travel along the found path and perform its operation at the goals [1]. However, planning problems originated from inspection or surveillance missions also include a problem of determining the goal locations. Thus, the problem is to determine the most suitable locations according to the mission objective while the path connecting them will be feasible and cost effective.

In inspection or surveillance missions, the goals are locations from which an object of interest is measured using the sensoric system, which has usually limited range, and therefore, the mission task can be formulated as a variant of the robotic coverage problem. An applicable approach to address the coverage problem is to decompose the problem into an independent determination of the sensing locations and the consecutive multi-goal path planning [2], [3] that connects the locations providing the required coverage. For view planning applications where it is necessary to consider both the motion and sensing costs [4], the sensing locations should be selected simultaneously with the path planning, otherwise a poor solution would be found [5].

However, even an independent determination of the minimal number of sensing locations is computationally demanding [6], as the problem can be formulated as a variant of the art gallery problem that is NP-hard for polygonal maps. The sequencing part of the problem is a variant of the traveling salesman problem (TSP), which is also NP-hard. Thus, considering all possible sensing locations in the sequencing part of the planning is computationally intractable and regular branch-and-bound or state space search methods are not applicable because of very large search space. Therefore, approximate algorithms are preferred and sampling based methods are utilized to determine possible sensing locations considering particular visibility constraints [7], [8] for which a connecting path can be found using efficient heuristics for the TSP, e.g., [9]. Here, it is worth to mention that planning a path connecting two locations in a high-dimensional configuration space is a challenging problem itself [10], and therefore, also in this part of the problem, sampling based approaches are usually considered for high-dimensional configuration spaces together with lazy path evaluation techniques [11].

In this paper, we aim to address the coverage planning problem of determining a feasible and cost effective path from which a given set of objects will be covered using a camera with a limited visibility range, i.e., we do not consider explicitly prescribed sensing locations. The problem is a variant of the covering salesman problem [12] that can be decomposed to the set cover problem and consecutive the TSP [13], [14]. Contrary to such approaches, we rather aim to simultaneously determine the suitable sensing locations together with optimization of the path.

The proposed approach is motivated by the recent approximate algorithm for the watchman route problem (WRP) based on the self-organizing neural network [15]. The main idea of the algorithm is that a path is represented by a neural network that evolves in the problem domain, where it is adapted towards not covered parts of the environment while the coverage from the path is maintained during the evolution. This approach requires a lot of visibility queries that can be computational expensive and to avoid this issue a supporting structure based on a cover set built on top of a triangular mesh is utilized in [15], which speeded up the process significantly and make it computationally feasible.

The main contribution of this paper is to provide a step further to extend the principles used in the algorithm developed for the WRP in a plane to a more general 3D environment, where visibility queries can be computationally demanding [16], [17]. Therefore, in this paper, we propose a simple algorithm to built a supporting structure providing information what can be covered from a single point and estimation of the space from which a particular object can be covered. The approach is based on dividing the free space into n elements and associating information about possible coverage of the objects. In particular, the query about possible coverage of objects from a particular element has complexity $O(1)$ and having an object to be covered, the query about possible covering regions has complexity $O(k)$, where k is the size of the query output. It is worth

Jan Faigl is with dept. of Computer Science and Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic `faiglj@fel.cvut.cz`

to mention that the aim of the proposed structure is not to provide precise information about visibility according to all visibility constraints, e.g., a limited field of view. It should be rather considered as a supporting structure to restrict the search space where suitable sensing locations can be found.

Although visibility queries are part of various 3D frameworks and it is a fundamental problem in computer graphics, there is not a similar structure (to the best of our knowledge) and the frameworks are not directly applicable unless a kind of ray-tracing technique is used, which seems to be too slow for the proposed approach.

The proposed planning method is mostly similar to [14]. We also consider a general representation of the environment, where 3D objects are tessellated into a triangular mesh, where particular triangles have to be covered from the inspection path being found. Our approach mainly differs in the way how the sensing locations are determined. In [14], the sensing is considered at discrete locations, which are determined prior finding a path connecting the needed locations. In our approach, we consider a larger space (but still restricted) where sensing locations are determined during solution of the sequencing part of the planning. Thus, the proposed planning procedure can be considered as a determination of the path guaranteeing coverage of all goals. Then, the path can be further processed to determine particular discrete sensing locations, a.k.a. solution of the vision points problem. However, the determination of the locations is inherently included in the utilized process of neural network evolution. Hence, the sensing locations are found simultaneously with the path.

The paper is organized as follows. The problem definition providing an application context of the proposed speeding up structure is presented in Section II together with an overview of the proposed planning method. An algorithm to construct the supporting structure is presented in Section III. In Section IV, the proposed multi-goal path planning approach with the supporting structure is described with a use case of its application. Finally, concluding remarks are presented in Section V.

II. Problem Definition

Our motivational problem is planning a surveillance mission for a micro aerial vehicle (MAV) operating in a combined indoor/outdoor environment, where it is requested to periodically take a snapshot of Objects of Interest (OoIs); hence, the robot needs to avoid obstacles. It is worth to remind that in this paper we are concern the structure supporting the visibility (or coverage) queries. The aim of the structure is to restrict the search space for determining sensing locations that provide the requested coverage. Therefore, we consider several assumptions and an abstract formulation of the problem to make the description of the proposed structure and planning approach more straightforward.

The considered problem is planning an inspection path for a mobile robot equipped with a camera with the limited visibility range ρ that is requested to see a given set of OoIs. Here, as the first step of the proposed planning approach, we assume omnidirectional vision, e.g., realized by camera heads attached to the robot. The operational environment $\mathcal{W} \subset \mathbb{R}^3$ is represented by a set of vertices $\boldsymbol{V}_\mathcal{W}$ connected to triangles $\boldsymbol{T}_\mathcal{W}$ representing obstacles. Without lost of generality we assume the triangular mesh $\mathcal{T} = (\boldsymbol{V}_\mathcal{W}, \boldsymbol{T}_\mathcal{W})$ of the obstacle surfaces is sufficiently dense. Let $\mathcal{W}_f$ be the free space of $\mathcal{W}$, $\mathcal{W}_f \subset \mathcal{W}$. Each triangle considered in this paper is defined by a sequence of three vertices defining the triangle (surface) normal that is oriented towards $\mathcal{W}_f$. All objects to be covered form a set $\boldsymbol{M} \subseteq \boldsymbol{T}_\mathcal{W}$; so, the objects are represented by a set of triangles.

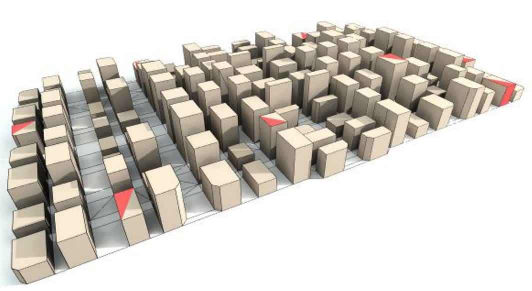
TABLE I
Used symbols

Symbol	Description
$\mathcal{W}$	the robot working space $\mathcal{W} \subset \mathbb{R}^3$
$\mathcal{C}$	the configuration space of the robot
ρ	the visibility distance range of the sensor (camera)
$\mathcal{W}_f \subseteq \mathcal{W}$	free space of $\mathcal{W}$
$\mathcal{T}$	a triangular mesh of obstacles of $\mathcal{W}$, $\mathcal{T} = (\boldsymbol{V}_\mathcal{W}, \boldsymbol{T}_\mathcal{W})$
$\boldsymbol{M}$	all parts of the objects' surfaces to be covered $\boldsymbol{M} \subseteq \boldsymbol{T}_\mathcal{W}$
o	the number of objects $o = \|\boldsymbol{M}\|$
m	an object to be covered $m \in \boldsymbol{M}$
G_{prm}	a graph representing roadmap $G_{prm} = (\boldsymbol{V}_{prm}, \boldsymbol{E}_{prm})$
$\boldsymbol{E}$	a tetrahedral mesh of $\mathcal{W}_f$
n	the number of tetrahedra in the mesh $n = \|\boldsymbol{E}\|$
$\boldsymbol{e}_i$	a tetrahedron of $\boldsymbol{E}$, $e_i \in \boldsymbol{E}$
$s_{\boldsymbol{e}}$	a surface (triangle) of the tetrahedron $\boldsymbol{e}$
C_m	a covering space of m with respect to ρ, $C_m \subseteq \mathcal{W}_f, C_m \subseteq \boldsymbol{E}$
$\boldsymbol{C}$	a union of all covering spaces $\boldsymbol{C} = \bigcup_{m \in \boldsymbol{M}} C_m$
$(\boldsymbol{e}, m, \boldsymbol{T})$	visibility dependency of $\boldsymbol{e}$ with respect to m, where $\boldsymbol{T}$ are surfaces of $\boldsymbol{e} \in \boldsymbol{E}$ with respect to $m \in \boldsymbol{M}$ supporting coverage of m from a point $p \in \boldsymbol{e}$
$\mathcal{V}_d$	a set of all visibility dependencies for each $\boldsymbol{e_i} \in \boldsymbol{E}$ with respect to m, $\mathcal{V}_d(m) = \{(\boldsymbol{e}_1, m, \boldsymbol{T}), \dots, (\boldsymbol{e}_n, m, \boldsymbol{T})\}$

We consider a mobile robot capable of moving in 3D environment (e.g., MAV) and for its point-to-point motion planning we consider the notion of the configuration space $\mathcal{C}$ and the Probabilistic Road Map (PRM) planner [18].

For each object $m \in \boldsymbol{M}$ we define a covering space $C_m \subseteq \mathcal{W}_f$ from which m can be seen using the sensor with the distance range ρ. C_m is found by the algorithm proposed in Section III. Although $\mathcal{C}$ can be a high dimensional configuration space, we consider the visibility in 3D, and therefore, we divide $\mathcal{W}_f$ into a set of tetrahedra considering the triangular mesh $\mathcal{T}$, e.g., using [19]. We assume that for each tetrahedron the ratios of the lengths of tetrahedron's edges is as small as possible, which pragmatically means the tetrahedral mesh is sufficiently dense and the centroid of each tetrahedron is inside the tetrahedron. We consider the tetrahedral mesh $\boldsymbol{E}$ as a set of tetrahedra $\boldsymbol{e}$, where each $\boldsymbol{e}$ consists of four triangles, without a formal introduction of tetrahedral mesh vertices and triangles to make the text more readable. The particular point of our interest are tetrahedra incident with $m \in \boldsymbol{M}$ as such a tetrahedron is the initial step for building C_m. An example of visualization of the environment $\mathcal{W}$ and objects $\boldsymbol{M}$ is shown in Fig. 1.

$\mathcal{C}$ is linked with $\mathcal{W}$ using the centroids of the tetrahedra $\boldsymbol{E}$ and vertices of the roadmap find by the PRM method. For simplicity, we consider centroids as the initial configurations for building the roadmap, instead of a random sampling $\mathcal{C}$.


(a) environment and objects of interest – $\mathcal{W}$, $\boldsymbol{M}$

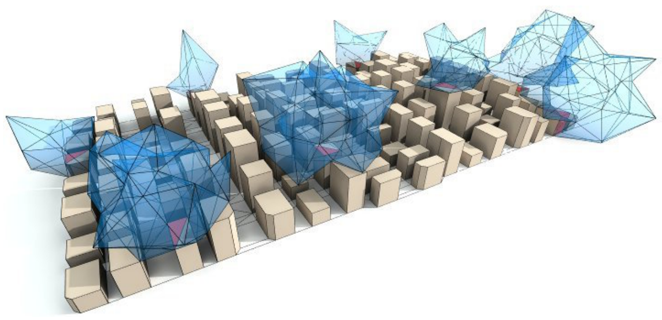

(b) covering spaces – $\boldsymbol{E}$

Fig. 1. Example of the environment representation, tetrahedralization of the freespace, objects of interest (red triangles) and their covering spaces. The red triangles denote the particular objects (part of them) of interest.

Then, we discard all tetrahedra without connected centroids with the roadmap; hence, for each tetrahedron, we have at least one feasible configuration. In the rest of the paper, we assume that all $\boldsymbol{e} \in \boldsymbol{E}$ have at least one feasible configuration and there exists a feasible path to other tetrahedra (their associated configurations). The built roadmap forms a graph $G_{prm} = (\boldsymbol{V}_{prm}, \boldsymbol{E}_{prm})$, where each vertex $v \in \boldsymbol{V}_{prm}$ is associated to a configuration $c \in \mathcal{C}_{free}$, which projection to $\mathbb{R}^3$ is the centroid of the particular $\boldsymbol{e} \in \boldsymbol{E}$.

The used notation is depicted in Table I. Having the above defined preliminaries, the planning problem can be defined as follows. *Find the shortest closed inspection path $\boldsymbol{I}$ in the graph G_{prm}, $\boldsymbol{I} = (v_1, v_2, \ldots v_k), v_1 = v_k, v_i \in G_{prm}$ such that all objects $\boldsymbol{M}$ will be seen from $\boldsymbol{I}$ by the sensor with the visibility range ρ, i.e., for each $m \in \boldsymbol{M}$ there exists $v_i \in \boldsymbol{I}$ such that $v_i \in C_m$.*

A. Planning Method Overview

The planning problem is basically a selection of configurations from $\boldsymbol{V}_{prm}$ that are in the covering spaces $\boldsymbol{C}$ such that the path connecting them is the shortest one, which is the problem addressed by the self-organizing map (SOM) for the WRP [15] and later used for finding the path connecting a set of convex goals in 2D [20]. Considering such a selection technique is available, the planning approach can be summarized in the following steps:

1) Tetrahedralization of the working environment (e.g., using [19]);
2) Generation of the motion planning roadmap (e.g., using the PRM method [18]);
3) Construction of covering spaces C_m for each $m \in \boldsymbol{M}$;
4) Determination of the inspection path using SOM;

The first two steps are straightforward. The construction of the covering spaces is presented in the next section and a brief description how the SOM technique can be utilized is presented in Section IV.

III. Construction of Covering Spaces

The covering space C_m of the object m consists of a set of tetrahedra and represents a part of $\mathcal{W}_f$ from which the whole object m can be seen with respect to the limited visibility range ρ. Thus, for each m we are looking for tetrahedra C_m such that distance of vertices of each tetrahedron $\boldsymbol{e} \in C_m$ from the vertices of m is less or equal to ρ. However, we have to deal with obstacles, as for each point of C_m the visibility of the whole m must be guaranteed. The proposed construction algorithm for determining C_m is based on an iterative procedure that inserts a new tetrahedron $\boldsymbol{e}$ to C_m while the fundamental constraint of the visibility of m from $p \in \boldsymbol{e}$ is preserved.

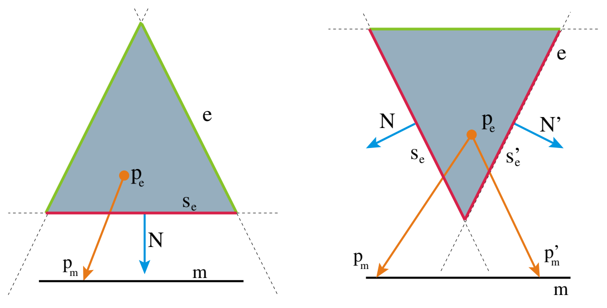


Fig. 2. Examples of the visibility dependency of the tetrahedron $\boldsymbol{e}$ with respect to m; (a) $(\boldsymbol{e}, m, \boldsymbol{T}) = \{s_e\}$; (b) $(\boldsymbol{e}, m, \boldsymbol{T}) = \{s_e, s'_e\}$.

The visibility means that for any point $p \in \boldsymbol{e}$ the ray connecting p with any point of m does not intersect an obstacle. Hence, the key point for keeping the visibility of m from $\boldsymbol{e}$ are surfaces of $\boldsymbol{e}$ that can be intersected by such a ray. We denote such surfaces as $\boldsymbol{T} \subset \boldsymbol{e}$ with respect to m and define a notion of visibility dependency $(\boldsymbol{e}, m, \boldsymbol{T})$, which assign $\boldsymbol{T}$ to $\boldsymbol{e}$ regarding m. For each $m \in \boldsymbol{M}$ and tetrahedron $\boldsymbol{e} \in \boldsymbol{E}$ we find the set of surfaces $\boldsymbol{T}$ by testing if a vertex of m is on the opposite side of the plane defined by $s_{\boldsymbol{e}} \in \boldsymbol{e}$ than a point inside $\boldsymbol{e}$. The test is performed as a scalar product of the $s_{\boldsymbol{e}}$ normal and ray (p_e, p_m). The principle is schematically depicted in Fig. 2. The complexity of determination of all visibility dependencies is $\Theta(no)$, where n is the number of tetrahedra ($n = |\boldsymbol{E}|$) and o is the number of objects $o = |\boldsymbol{M}|$.

Having the visibility dependencies $(\boldsymbol{e}, m, \boldsymbol{T})$ for each $\boldsymbol{e} \in \boldsymbol{E}$ and $m \in \boldsymbol{M}$, the building of C_m is relatively straightforward as we basically consider each tetrahedron $\boldsymbol{e}$ and test if the visibility constraint will be satisfied after insertion of $\boldsymbol{e}$ into C_m. However, the tetrahedra can be inserted in an arbitrary order, and therefore, we define a notion of transitive dependency of $\boldsymbol{e}$ on other tetrahedra that must be inserted to

Algorithm 1: Construction of covering space C_m

```
Input: $m$ – an object to be covered
Input: $\boldsymbol{E}, \mathcal{V}_d(m)$ – tetrahedral mesh and all visiblity
       dependencies with respect to $m$
Output: $C_m$ – the covering space for the object $m$
1 $\boldsymbol{e}_{obj} \leftarrow$ get $\boldsymbol{e}$ such that $\boldsymbol{e} \in \boldsymbol{E} \wedge s \in \boldsymbol{e} \wedge \mathrm{incident}(s, m)$;
2 $C_m \leftarrow \{\boldsymbol{e}_{obj}\}$;
3 $E_{free} \leftarrow \boldsymbol{E} \setminus C_m$;
4 $E_{close} \leftarrow \emptyset$;
5 while $\exists \boldsymbol{e} \in \boldsymbol{E}_{free} \wedge \boldsymbol{e}_{nbr} \in C_m \wedge \mathrm{incident}(\boldsymbol{e}, \boldsymbol{e}_{nbr})$ do
6     $G(\boldsymbol{E}_G, \boldsymbol{H}) \leftarrow \mathrm{get\_dependency}(m, C_m, (\boldsymbol{e}, m, \boldsymbol{T}))$;
7     $C_m \leftarrow \mathrm{add\_tetrahedra}(C_m, G(\boldsymbol{E}_G, \boldsymbol{H}))$;
```

C_m in order to satisfy the visibility of m from $\boldsymbol{e}$. So, during the iterative insertion we construct an auxiliary graph G with information about the transitive dependency. In addition, we maintain two sets $\boldsymbol{E}_{close}$ containing tetrahedra transitively dependent on obstacle or closed tetrahedron and $\boldsymbol{E}_{free}$ with all not yet processed tetrahedra. For the incremental construction of C_m we need a relation of neighbouring tetrahedra (or incident triangles), and therefore, we define the Boolean operator $\mathrm{incident}(\boldsymbol{e}_1, \boldsymbol{e}_2)$ that is true if $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ share the same triangle (except its orientation) and false otherwise. The construction procedure is depicted in Algorithm 1 and the sub-procedures $\mathrm{get\_dependency}$ and $\mathrm{add\_tetrahedra}$ in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: Construction of the dependency graph G

```
Input: $m$ – an object to be covered
Input: $\boldsymbol{e}_0$ – tetrahedron being added to $C_m$
Input: $\boldsymbol{E}, \mathcal{V}_d(m)$ – tetrahedral mesh and all visiblity
       dependencies for $\boldsymbol{e}_i \in \boldsymbol{E}$ with respect to $m$
Output: $G(\boldsymbol{E}_G, \boldsymbol{H})$ – the dependency graph for $\boldsymbol{e}_0$
1 $\boldsymbol{E}_{tmp} \leftarrow \{\boldsymbol{e} | \boldsymbol{e} \in \boldsymbol{E} \wedge \mathrm{incident}(\boldsymbol{e}_0, \boldsymbol{e})\} \cup \{\boldsymbol{e}_0\}$;
2 $\boldsymbol{H} \leftarrow \emptyset$;
3 while $\boldsymbol{E}_{tmp} \cap \boldsymbol{E}_{free} \neq \emptyset$ do
4     $\boldsymbol{e}_i \leftarrow$ get $\boldsymbol{e}$ such that $\boldsymbol{e} \in \boldsymbol{E}_{tmp} \wedge \boldsymbol{e} \in \boldsymbol{E}_{free}$;
5     $\boldsymbol{E}_{free} \leftarrow \boldsymbol{E}_{free} \setminus \{\boldsymbol{e}_i\}$;
6     $\boldsymbol{T}_{neigh} \leftarrow \{t | t \in (\boldsymbol{e}_i, m, \boldsymbol{T}) \wedge \mathrm{incident}(\boldsymbol{e}_i, t)\}$;
7     if $\boldsymbol{T}_{neigh} \cap \boldsymbol{T}_{\mathcal{W}} \neq \emptyset$ then
8         $\boldsymbol{E}_G \leftarrow \boldsymbol{E}_G \cup \{\boldsymbol{e}_i\}$ // $\boldsymbol{e}_i$ is incident with obstacle;
9         $\boldsymbol{E}_{close} \leftarrow \boldsymbol{E}_{close} \cup \{\boldsymbol{e}_i\}$;
10    else
11        for $\boldsymbol{e}_{neigh} \in \{\boldsymbol{e} | \boldsymbol{e} \in \boldsymbol{E} \wedge t \in \boldsymbol{T}_{neigh} \wedge t \in \boldsymbol{e}\}$ do
12            $\boldsymbol{E}_{tmp} \leftarrow \boldsymbol{E}_{tmp} \cup (\boldsymbol{E}_{free} \cap \{\boldsymbol{e}_{neigh}\})$;
13            $\boldsymbol{E}_G \leftarrow \boldsymbol{E}_G \cup \{\boldsymbol{e}_{neigh}\}$;
14            $\boldsymbol{H} \leftarrow \boldsymbol{H} \cup \{(\boldsymbol{e}_i, \boldsymbol{e}_{neigh})\}$ // add the relation;
15 $G \leftarrow G(\boldsymbol{E}_G, \boldsymbol{H})$;
```

In Algorithm 2, the auxiliary graph $G = (\boldsymbol{E}_G, \boldsymbol{H})$ is determined. G represents all tetrahedra on which $\boldsymbol{e}$ being added to C_m is transitively dependent, i.e., each vertex of G represents a tetrahedron and an oriented edge $h \in \boldsymbol{H}$ between $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ indicates the transitive dependency of $\boldsymbol{e}_1$ on $\boldsymbol{e}_2$. The tetrahedron $\boldsymbol{e}_0$ can be added to C_m only if it is not dependent on a tetrahedron that is in $\boldsymbol{E}_{close}$ or it is not incident with an obstacle. The procedure is repeated until all transitively dependent tetrahedra are processed (Line 3). Then, G is used for adding all tetrahedra that are not incident with an obstacle or closed tetrahedron into C_m using the procedure $\mathrm{add\_tetrahedra}$, see Algorithm 3.

Algorithm 3: Adding tetrahedra to C_m

```
Input: $m$ – an object to be covered
Input: $G(\boldsymbol{E}_G, \boldsymbol{H})$ – the current dependency graph
Output: $C_m$ – the covering space for the object $m$
1 $\boldsymbol{E}_{act} \leftarrow \boldsymbol{E}_{close}$                // test all close tetrahedra;
2 while $|\boldsymbol{E}_{act}| > 0$ do
3     $\boldsymbol{e}_{act} \leftarrow$ get a tetrahedron from $\boldsymbol{E}_{act}$;
4     $\boldsymbol{E}_{close} \leftarrow \boldsymbol{E}_{close} \cup \{\boldsymbol{e}_{act}\}$;
5     $\boldsymbol{E}_{act} \leftarrow \boldsymbol{E}_{act} \setminus \{\boldsymbol{e}_{act}\}$;
6     $\boldsymbol{E}_{dep} \leftarrow \{\boldsymbol{e} | \boldsymbol{e} \notin \boldsymbol{E}_{close} \wedge (\boldsymbol{e}, \boldsymbol{e}_{act}) \in \boldsymbol{H}\}$;
7     $\boldsymbol{E}_{act} \leftarrow \boldsymbol{E}_{act} \cup \boldsymbol{E}_{dep}$;
8 $C_m \leftarrow C_m \cup (\boldsymbol{E}_G \setminus \boldsymbol{E}_{close})$ // add non closed dep. tet.;
```

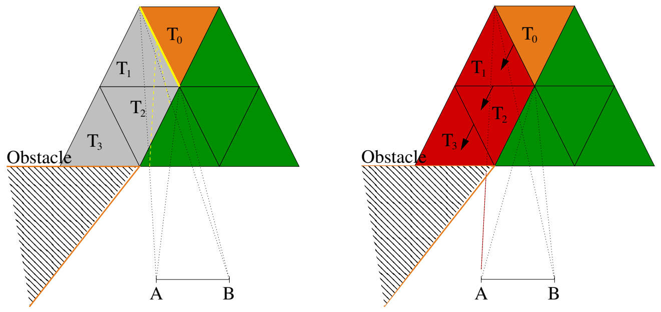


Fig. 3. An example of the transitive dependency, triangles represent surfaces' of tetrahedra. The orange tetrahedra are tested for being added to C_m, the green triangles are tetrahedra already in C_m, and gray triangles are not yet processed tetrahedra. The red triangles are tetrahedra from, which visibility to segment (AB) of m cannot be guaranteed. In the left figure, the tetrahedra can be added to C_m while for the right case, T_0 is transitively dependent on T_3 with respect to m and visibility of m from T_0 cannot be guaranteed because T_3 is incident with an obstacle.

An example of the transitive dependency is shown in Fig. 3 using triangles (2D slice of tetrahedron) for a clarity. Although the proposed construction of the covering space C_m is only approximation, the crucial point of C_m is that it guarantees coverage of m within the limited sensor range ρ.

A. Computational Complexity and Queries

The computational complexity of Algorithm 2 is $O(n)$ as in the worst case all tetrahedra can be processed. It is also the case of Algorithm 3 where up to $O(n)$ elements can be examined. Thus, the total complexity of the construction algorithm is $\Theta(no)$.

Regarding a usage of the structure, it provides $O(1)$ queries for test if a point in a particular tetrahedron can cover an object. For a general point, the complexity depends on the finding the particular tetrahedron for the point, which can be based on the kd-tree using centroids of the tetrahedra. Such a query can be answered in the average complexity $O(\log n)$.

TABLE II

CONSTRUCTING TIMES FOR TETRAHEDRAL MESH COVERING SPACES

Parameter	Scene 1	Scene 2	Scene 3	Scene 4
No. of triangles	1280	1280	2444	2444
No. of tetrahedra	2 538	10 852	27 071	113 505
Tetrahedral mesh [ms]	90	351	809	3 705
Covering spaces [ms]	33	46	271	746

However, the closest centroid does not guarantee the point is inside the particular tetrahedron for a weakly constructed tetrahedral mesh. Then, for such a case a local search method can be applied. On the other hand, the proposed planning approach considers configurations with associated information about its tetrahedron; hence, it is not necessary to perform the finding query and the information about covering is available instantly.

We consider four scenes with different number of tetrahedra to provide an overview of the real computational requirements using a computer with iCore7 3.4 GHz CPU. Scene parameters and construction times are depicted in Table II. In all cases, 10 covering spaces are constructed.

In addition, we consider 100 000 random queries in the Scene 2, where the average query time to find a tetrahedron for a random point is about 94 μs using an exhaustive search.

IV. MULTI-GOAL INSPECTION PLANNING

The planning algorithm for determining the inspection path $\boldsymbol{I}$ is inspired by the self-organizing map (SOM) approach for the WRP [15], which is based on SOM adaptation schema for the TSP [21]. However, rather than SOM for the polygonal domain, we utilize a SOM variant for a graph input [22] and use the roadmap G_{prm} as the graph.

The planning algorithm is basically an unsupervised learning procedure for two-layered competitive neural network. An input vector is a vertex of G_{prm} representing an object being covered. The output layer consists of k units representing neuron weights. The output units are organized into a unidimensional structure $\mathcal{N} = (\nu_1, \ldots, \nu_k)$, which is called a ring in the rest of this paper. Hence, the ring represents a path evolving in the graph G_{prm}. The weights are coordinates in $\mathcal{C}$; however, restricted to the roadmap G_{prm}.

The learning process is an iterative procedure in which objects $m \in \boldsymbol{M}$ are presented to the network in a random order. For simplicity, we can assume that each object m has associated a particular vertex $v_m \in \boldsymbol{V}_{prm}$. First, a winning neuron ν^* with the shortest distance to v_m is determined in the competitive phase. Then, ν^* together with its neighbouring nodes are adapted towards v_m. The adaptation is terminated if each object has its winner neuron in a sufficiently small distance. Using the graph input, the learning process can be described as an evolution of a path (ring) in the problem domain defined by the G_{prm}. The adaptation of nodes can be interpreted as their movement towards the presented goal along paths found in the G_{prm}, e.g., using Dijkstra's algorithm. The neurons' weights represent the inspection path $\boldsymbol{I}$ and the order of object visits can be retrieved by traversing the output layer (ring).

The procedure is relatively simple but its simplicity provides a great flexibility to address variant of inspection planning. First, we introduce usage of the covering space C_m. The modification of the standard TSP algorithm is straightforward. In the competitive phase, we consider C_m to select not only the winning neuron ν^* but also a new goal towards which the winner will be updated; thus, the competitive rule is as follows

$$(\nu^*, p) \leftarrow \mathrm{argmin}_{\nu \in \mathcal{N}, \boldsymbol{e} \in C_m}(\mathrm{distance}(\nu, \mathrm{centroid}(\boldsymbol{e})),$$

where p is a point of the particular $\boldsymbol{e}$, e.g., the centroid of $\boldsymbol{e}$. Then, ν^* and its neighbouring neurons are adapted towards p instead of v_m. The idea is that instead of direct planning to visit the object, we rather prefer to find some point from which the object will be covered. Here, we employ the constructed covering space C_m, which allows the proposed straightforward extension without any performance lost related to compute the coverage.

Notice, it is not necessary to consider all tetrahedra in C_m for evaluation of the best possible goal p. We can sample only few of them or we can use other techniques. The advantage of C_m is that it provides explicit representation of possible goal candidates from which the object can be covered.

A. Use Case

The feasibility of the proposed planning approach based on the covering spaces C_m has been verified in a city like environment represented by 2 444 triangles, see Fig. 1. We consider restricted visibility range ρ=2. The objects to be covered are 10 triangles representing 7 physical objects of interest. One object is represented by 3 incident triangles that is located at the bottom right part of the environment. Two underlying graphs with 27 071 and 113 505 nodes (for two tetrahedral meshes) are used to provide an overview of real computational requirements.

TABLE III

COMPUTATIONAL TIMES OF PARTICULAR PLANNING PARTS

Tet. mesh	PRM	Covering spaces	SOM epoch=20	SOM epoch=40	SOM epoch=60
0.8 s	0.7 s	210 ms	0.6 min	1.1 min	1.4 min
3.7 s	2.8 s	352 ms	3.3 min	6.5 min	8.4 min

Particular required computationally times of each part of the proposed planning method are depicted in Table III for a single core CPU running at 3.4 GHz. The final solution has been found after 60 learning epoch of the SOM learning procedure. However, SOM provides a solution at the end of each epoch and for example after 40 epochs the path length was 38.6 while the final solution after 60 epoch has length 27.3. In the case of a finer tetrahedral mesh (with 113 505 tetrahedra) the lengths are 35.2 and 26.3, respectively. An example of found solution is visualized in Fig. 4.

During the learning, particular distances and paths between nodes are computed on demand and saved for a latter usage, therefore, the complete distance matrix is not

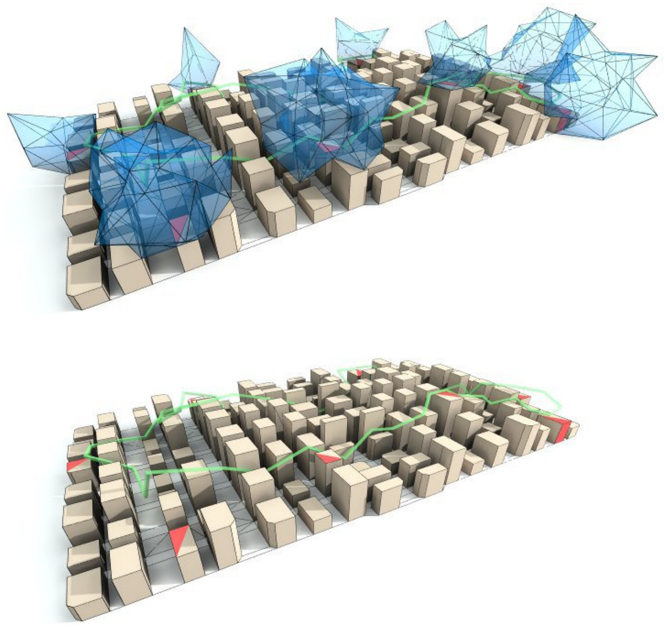


Fig. 4. An example of found solution in the city-like scenario.

computed. The memory footprint is about 1 GB and 10 GB, respectively, which is significantly lower than a required storage for the complete distance matrix. It should be noted that herein presented results are part of the feasibility study of the proposed covering space structure and the planning method, and we consider the distance matrix for simplicity.

V. Conclusion

In this paper, we present an algorithm for construction supporting structures to avoid computing covering queries in the proposed 3D inspection planning and demonstrate how it can be utilized in multi-goal inspection planning. The structure can be considered as an enabling technique for applying self-organizing map (SOM) based planning principles in 3D environments. SOM provides interesting results for inspection planning in planar environments where they are able to solve multi-goal path planning problems with point or polygonal goals [20] and even problems without explicitly prescribed goals [15].

The proposed structure of covering spaces is our early results towards applying self-organizing principles in a high-dimensional space. Although the current problem formulation assume an omnidirectional vision, the proposed principles provide a different mechanism of searching the state space than regular branch and bound algorithms or decomposed approaches. It allows a simultaneous determination of suitable goal locations during solving the sequencing part of the multi-goal path planning. Therefore, additional constraints can be considered during the self adaptation of the used neural network, e.g., a coverage from the ring can be computed according to the robot's orientation. Hence, we expect the proposed framework will allow to consider not only sensor with a limited field of view, but additional motion constraints of the robot. In addition, it is not necessary to precompute roadmap using centroids of all tetrahedra. Lazy motion planning such as [14] can be combined with the covering spaces. Such further developments are subject of our current work.

Acknowledgments

The work of J. Faigl was supported by the Czech Science Foundation (GAČR) under research project No. 13-18316P.

References

[1] S. N. Spitz and A. A. G. Requicha, "Multiple-Goals Path Planning for Coordinate Measuring Machines," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 2322–2327.

[2] F. Zhao, X. Xu, and S. Xie, "Computer-aided inspection planning-the state of the art," *Computers in Industry*, vol. 60, no. 7, pp. 453–466, 2009.

[3] T. Danner and L. E. Kavraki, "Randomized Planning for Short Inspection Paths," in *Proceedings of The IEEE International Conference on Robotics and Automation (ICRA)*. San Fransisco, CA: IEEE Press, April 2000, pp. 971–976.

[4] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Comput. Surv.*, vol. 35, no. 1, pp. 64–96, 2003.

[5] P. Wang, "View planning with combined view and travel cost," Ph.D. dissertation, Simon Fraser University, 2007.

[6] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: maximizing information while minimizing communication cost," in *The Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, 2006, pp. 2–10.

[7] H. H. Gonzalez-Banos and J.-C. Latombe, "Navigation Strategies for Exploring Indoor Environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[8] J. Faigl, M. Kulich, and L. Přeučil, "A sensor placement algorithm for a mobile robot inspection planning," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 3-4, pp. 329–353, 2011.

[9] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic," *European Journal of Operational Research*, vol. 126, no. 1, 2000.

[10] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, May 2006.

[11] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning Tours of Robotic Arms among Partitioned Goals," *Int. J. Rob. Res.*, vol. 25, no. 3, pp. 207–223, 2006.

[12] J. R. Current and D. A. Shilling, "The covering salesman problem," *Transportation Science*, vol. 23, no. 3, 1989.

[13] P. S. Blaer and P. K. Allen, "View planning and automated data acquisition for three-dimensional modeling of complex sites," *J. Field Robot.*, vol. 26, no. 1112, pp. 865–891, Nov. 2009.

[14] B. Englot and F. Hover, "Planning complex inspection tasks using redundant roadmaps," in *15th International Symposium of Robotics Research (ISRR)*, Flagstaff, AZ, August 2011.

[15] 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.

[16] F. Durand, G. Drettakis, and C. Puech, "The 3D Visibility Complex," *ACM Transactions on Graphics*, vol. 21, no. 2, pp. 176–206, 2002.

[17] M. Nouri Bygi and F. Ghodsi, "3d visibility and partial visibility complex," in *International Conference on Computational Science and its Applications, (ICCSA)*, aug. 2007, pp. 208–207.

[18] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[19] H. Si, "Tetgen - a quality tetrahedral mesh generator and a 3d delaunay triangulator," [cit. 09-05-2012]. [Online]. Available: http://tetgen.berlios.de

[20] J. Faigl, V. Vonásek, and L. Přeučil, "Visiting convex regions in a polygonal map," *Robotics and Autonomous Systems*, 2012, (*accepted, paper in press*), http://dx.doi.org/10.1016/j.robot.2012.08.013.

[21] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *Journal of the Operational Research Society*, pp. 919–928, 1997.

[22] T. Yamakawa, K. Horio, and M. Hoshino, "Self-Organizing Map with Input Data Represented as Graph," in *Neural Information Processing*. Springer Berlin / Heidelberg, 2006, pp. 907–914.