The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

# Distributed Coverage Control on Surfaces in 3D Space

Andreas Breitenmoser, Jean-Claude Metzger, Roland Siegwart and Daniela Rus

*Abstract*— **This paper addresses the problem of deploying a group of networked robots on a non-planar surface embedded in 3D space. Two distributed coverage control algorithms are presented that both provide a solution to the problem by discrete coverage of a graph. The first method computes shortest paths and runs the Lloyd algorithm on the graph to obtain a centroidal Voronoi tessellation. The second method uses the Euclidean distance measure and locally exchanges mesh cells between approximated Voronoi regions to reach an optimal robot configuration. Both methods are compared and evaluated in simulations and in experiments with five robots on a curved surface.**

## I. INTRODUCTION

Many multi-robot applications require a robot team to be distributed in an environment for efficient performance of service or monitoring tasks. Usually, coverage algorithms are studied in the context of simple 2D environments, in which a group of networked robots deploys in the plane. A great potential lies in methods that extend planar coverage to coverage of sufaces embedded in 3D space. Distributed coverage on non-planar surfaces is useful for a broad class of real-world applications, such as multi-robot exploration in rough terrain with slopes and varying traversability, or collaborative inspection of industrial structures by a team of climbing robots.

We address the distributed coverage of such non-planar surfaces in this paper. The two proposed coverage control algorithms both divide the surface by a centroidal Voronoi tessellation (CVT) and distribute the robots in an optimal placement on the surface in 3D space (see Figure 1). The algorithms are executed directly on a graph and are discretized versions related to the Voronoi coverage method introduced in [1]. This basic method has further been studied in [2], [3], [4] for robotic coverage of two- and three-dimensional convex space and more generally works in convex environments of $N$ dimensions. However, the coverage of a surface embedded in 3D space is no straightforward extension to higher dimensions and in fact presents a constrained optimization problem. The Voronoi region is part of the surface and its centroid must be constrained to the surface. In [5], a constrained CVT is computed on non-planar surfaces by projecting the centroids of the Voronoi regions onto the surface in each iteration step. This approach assumes knowledge of the analytical expression for the parametric surface and is limited to simple standard geometries, though. Voronoi coverage on a non-planar surface is related to the problem of covering a planar non-convex environment. In the latter case, the obstacles in the environment introduce constraints and the non-convexity of the environment again makes it hard to find general solutions for the Voronoi coverage method (see [2], [6], [7], [8] for creative approaches to the problem).

Fig. 1. Distributed coverage of a curved surface with hole. Five e-puck robots run Algorithm 2 and cover the non-planar non-convex surface (left: top view with surface mesh and Voronoi partition overlay, right: side view).

In this paper, we use a mesh to represent the environment and the distributed coverage algorithms work on this graph in the discrete domain. Coverage on a graph results in approximated solutions but resolves problems that arise from topological constraints (e.g. curved surfaces, obstacles in non-convex environments), as a graph is an abstract representation. Moreover, relying on graphs when developing coverage control strategies allows us to benefit from findings in graph theory and computer graphics.

In computer graphics, Voronoi tessellations are a widely used concept in the context of 3D surface mesh generation. 3D surface meshes are most commonly triangle meshes, which are either generated manually with a CAD tool or automatically from 3D point clouds (see [9], [10] and references therein). In terms of robotics, the meshes represent maps of the environment and are either known a priori or

This work was done in the Distributed Robotics Laboratory at MIT. This work was supported by ALSTOM, and in part by the ARO MURI SWARMS grant 544252, the ARL CTA MAST grant 549969, the ONR MURI SMARTS grant N0014-09-1051, NSF grants IIS-0513755, IIS-0426838, EFRI-0735953, and The Boeing Company.

Andreas Breitenmoser, Jean-Claude Metzger and Roland Siegwart are with the Autonomous Systems Laboratory (ASL), ETH Zurich, Tannenstrasse 3, 8092 Zurich, Switzerland, andreas.breitenmoser@mavt.ethz.ch, jemetzge @ethz.ch, rsiegwart@ethz.ch

Daniela Rus is with the Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, Cambridge, MA 02139, USA, rus@csail.mit.edu

retrieved by a robot online with a laser range finder for example. A method for isotropic remeshing of triangulated surfaces is proposed in [11]. The surfaces are parameterized and mapped into a planar parameter space, in which the CVT is calculated. This technique requires cutting of the mesh before mapping if the mesh is closed or of genus $> 0$, which presents a problem of its own and furthermore complicates the application of the method on robots. In [12], mesh partitioning based on geodesic centroidal tessellation is used to segment the surface, whereas the CVT is approximated and applied to coarsening of surface meshes in [13]. The two methods from [12] and [13] offer interesting ideas for robot coverage control. We adapted some elements from these centralized methods to develop the two decentralized coverage control algorithms of this paper.

Meshes or grid representations are also extensively used in robotics. As a prominent example, occupancy grids present a well-known model in robot mapping and exploration [14]. With respect to coverage control and Voronoi coverage in particular, the work in [15] is the first that proposes Voronoi coverage on a graph representing a discretized environment. We have independently followed a similar approach using the discretization of Voronoi partitioning in our algorithms. However, we target a different problem of covering non-planar surfaces. Moreover, we verify our approach with a group of robots in physical experiments.

The paper is organized as follows. The problem of generating Voronoi partitions on a surface mesh is formulated in Section II. Two different coverage control algorithms that both lead to coverage of surfaces in 3D space are described in Section III. Simulations on surface meshes with varying complexity and number of robots demonstrate the functionality of the two algorithms in Section IV. Then, Section V presents results from experiments on a curved surface with a real robot team. Finally, conclusions are given in Section VI.

## II. Problem Formulation

A group of $n$ networked robots must be distributed on a non-planar surface, i.e. a 2D manifold embedded in $\mathbb{R}^3$. Let $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n \in \mathbb{R}^{3n}$ be the positions of the robots in the continuous domain, $\mathbf{P}^* = [\mathbf{p}^*{}_i]_{i=1}^n \in \mathbb{R}^{3n}$ the constrained positions of the robots on a graph embedded in $\mathbb{R}^3$ and $P^* = \{p_1^*, ..., p_n^*\}$ the set of robots' position nodes on the graph. The surface is approximated by a polygon mesh and represented by an undirected graph $G = (Q, E)$ with node set $Q$ and edge set $E = Q \times Q$.

Voronoi coverage from [1] works in the continuous domain and optimizes the robot configuration by minimizing

$$\mathcal{H}_{con}(\mathbf{P}) = \sum_{i=1}^{n} \mathcal{H}_{con_i}(\mathbf{p}_i) = \sum_{i=1}^{n} \int_{V_i} D(\mathbf{p}_i,\, \mathbf{x})^2 \;\phi(\mathbf{x})\, d\mathbf{x}\ , \tag{1}$$

which results in a CVT, i.e. the generators for the Voronoi tessellation are themselves the centroids of the Voronoi regions $V_i$. The distance measure $D(\cdot)$ depends on the metric used (e.g. Euclidean distance). The CVT can be computed in a fixed point iteration by the Lloyd algorithm [16].

Next we formulate the CVT for the discrete case, making use of its equivalent in graph theory, the *graph Voronoi diagram* [17]. $Q$ is divided according to the Voronoi partitions $V_{1_i}^* = \left\{q \in Q \mid D(q,\, p_i^*) < D(q,\, p_j^*),\, \forall\, j \neq i\right\}$ of the graph $G$, where $j \in \{1, ..., n\}$ and $D(u,\, w)$ a distance measure; here the shortest path $d(u,\, w)$ between two nodes $u$ and $w$ on the graph is used. Nodes $q$ with equal distance from two generators are assigned to a partition according to the priority given to the robots.

Each Voronoi region can alternatively be defined by a union of several mesh cells $C_k$, or respectively the set of nodes of $G' = (Q', E')$, the dual graph of $G$, which represents the cells of the polygon mesh. This leads to $V_{2_i}^* = \{q' \in Q' \mid C = \bigcup C_k(q')\}$ , where $q' \mapsto C_k(q')$ and the set $C$ approximates the continuous Voronoi region $V_i$.

Both formulations of Voronoi regions result in an approximation due to discretization. Resolution and accuracy are given by the cell size and misadjustments of the mesh may result in suboptimal performance of the overall approach. The cost from Equation (1) is discretized to

$$\mathcal{H}_{dis}(P^*) = \sum_{i=1}^{n} \mathcal{H}_{dis_i}(p_i^*) = \sum_{i=1}^{n} \sum_{v \in V_i^*} D(p_i^*,\, v)^2 \;\phi(v)\ . \tag{2}$$

The CVT is then obtained by running the Lloyd algorithm directly on the graph. For $v \in V_i^* = V_{1_i}^*$ the nodes $q$ of graph $G$, $D(p_i^*, v) = d(p_i^*, q)$ the shortest path and $\phi(q)$ the weight of the nodes, each generator converges to an *intrinsic center of mass*

$$\gamma_{V_{1_i}^*} = \underset{p_i^* \in V_{1_i}^*}{\operatorname{argmin}}\, \mathcal{H}_{dis_i}(p_i^*) \tag{3}$$

contained in the graph, which is a minimizer of the single cost $\mathcal{H}_{dis_i}(p_i^*)$ in (2) (see [12] for the continuous version).

An alternative approach for CVT computation, different from methods relying on the Lloyd algorithm, is described in [13]. This approach approximates the CVT by iteratively minimizing the cost function (2) using local boundary tests on the mesh cells only. In this case, $v \in V_i^* = V_{2_i}^*$ are nodes $q'$ of the dual graph $G'$, $D(p_i^*, v) = \|\mathbf{c}_k - \mathbf{p}_i^*\|$ is the Euclidean distance with $\mathbf{c}_k \in \mathbb{R}^3$ the centroid and $\phi(\mathbf{c}_k)$ the density of a cell $C_k(q')$. In order to minimize the cost in (2) and guarantee convergence to a CVT, the generators must be placed in the centroids of the Voronoi regions. We compute the centroids of each partition $V_{2_i}^*$ as $\gamma_{V_{2_i}^*} = \sum_{q' \in V_{2_i}^*} \mathbf{c}_k\, \phi(\mathbf{c}_k) / \sum_{q' \in V_{2_i}^*} \phi(\mathbf{c}_k)$ and substitute the generators $p_i^*$ in (2) by these centroids $\gamma_{V_{2_i}^*}$. In accordance with [13] we get

$$\mathcal{H}_{dis\_approx} = \sum_{i=1}^{n} \sum_{q' \in V_{2_i}^*} \|\mathbf{c}_k - \gamma_{V_{2_i}^*}\|^2 \;\phi(\mathbf{c}_k)\ . \tag{4}$$

As consequence, the cost function $\mathcal{H}_{dis\_approx}$ depends no more on the generators $p_i^*$ and can simply be minimized by a *local cell exchange* between neighboring Voronoi regions.

For each mesh cell $C_A$ in a Voronoi region $V^*_{2_A}$ that lies at a boundary to a cell $C_B$ of a neighboring Voronoi region $V^*_{2_B}$ a local test is executed. The change in the cost function (4) is calculated for the case that: 1) $C_A$ is added to $V^*_{2_B}$, 2) $C_B$ is added to $V^*_{2_A}$ or 3) no cell is exchanged across the boundary. The case resulting in the highest reduction of cost is selected and the Voronoi regions are updated accordingly. If a mesh cell at the boundary of a Voronoi region $V^*_{2_C}$ has not yet been assigned to any other Voronoi region, i.e. the cell is *free*, it is incorporated directly into $V^*_{2_C}$ without any calculation of cost. This way, the Voronoi regions grow into areas of unassigned mesh cells to partition the free surface, while the ongoing exchange of cells with other Voronoi regions helps to minimize the overall cost in (4).

***Remark 1:*** In this paper, we use a homogeneous triangle mesh but other types of polygon meshes could be applied without loss of generality (e.g. a grid map as in [15]).

***Remark 2:*** By exchanging triangle cells for minimization, the Voronoi regions of the final partition become equal in size. However, equitable partitioning does not result by nature when using the Lloyd algorithm. We do not make use of edge weights here. Edge weights different from unity offer a possibility to control the size and shape of the Voronoi regions.

## III. Two distributed algorithms for coverage control on non-planar surfaces

We assume here that the triangle mesh has been generated in advance and is available as input to the algorithms. However, we will investigate the problem under two variable assumptions: the robots operate in *synchronous* or *asynchronous* mode and the environment may be *known* or *unknown*. Asynchronous mode (as opposed to synchronous mode) means that the robots do not wait for the other robots in order to calculate the Voronoi regions and regions' centroids. If the environment is known, the robots will directly know all the nodes in the surface mesh (given by graph $G$). In an unknown environment, the robots are able to sense the surface and discover mesh nodes within a sensor range of $R_s$ as they move. The detected nodes are added to the already discovered subgraph $G_i \subset G$. The robots can further communicate and exchange information among each other within a communication range of $R_c$. If two robots are within $R_c$ and share at least one common node, i.e. their subgraphs $G_i$ and $G_j$ are connected, the nodes are exchanged and the subgraphs merged ($G_i \cup G_j$).

### A. Front propagation algorithm

The first coverage control algorithm minimizes the cost function (2) by computing the Lloyd algorithm in decentralized fashion. The graph Voronoi regions are obtained from shortest path calculation based on the propagation of a discrete wave front on the graph. For the propagations a label correcting method based on the Dijkstra algorithm after [18] is used. Each robot $i$ can either be in a `COMPUTE` or a `MOVE` state. When a robot has reached its current goal node, it enters the `COMPUTE` state and calculates the next goal node $p^*_{i_{NEW}}$. Then the robot switches to the `MOVE` state and drives from $\mathbf{p}^*_i$ to $\mathbf{p}^*_{i_{NEW}}$ while updating its position in the graph (see Algorithm 1).

**Algorithm 1** FRONT PROPAGATION

**Require:** Set of robots $i \in \{1, ..., n\}$ with initial positions $\mathbf{p}^*_{i_0}$ on the surface, and each robot $i$ provided with:
- surface mesh represented by graph $G$ (known env.) or subgraph $G_i$ (unknown env.)
- localization, sensing and communication with neighbor robots (synchronous or asynchronous mode)

**loop**
  **if** (state == `COMPUTE`) **then**
    Request neighbor positions $\mathbf{p}^*_j = \mathbf{p}^*_{j_{NEW}},\ j \neq i$
    Compute Voronoi region $V^*_{1_i}$ (Dijkstra algorithm)
    Calculate $p^*_{i_{NEW}}$ (= first node toward $\gamma_{V^*_{1_i}}$)
    Update state
  **else** {state == `MOVE`}
    Move to $\mathbf{p}^*_{i_{NEW}}$ and update $p^*_i$
    Update state
  **end if**
**end loop**

During runtime the robots keep track of the nodes in the graph using following data structures: a priority queue with a sorted candidate list $L_F$ of the nodes at the propagating wave front, a distance map $M(q)$ with the shortest paths from each node to the closest robot and an identity set $I(q)$ with the identity of that closest robot.

In the `COMPUTE` state the robot detects the positions $p^*_j$ of the neighboring robots constrained to the graph and executes one iteration of the Lloyd algorithm. Robot $i$ initiates a *front propagation* that proceeds until the wave front reaches a neighbor robot ($M(q)$ and $I(q)$ are generated). At that point the propagation is paused and a *back propagation* starts on the visited nodes. The back propagation terminates when the nodes at the front are going to be closer to robot $i$ than to its neighbor again (checking against $M(q)$). All the nodes which are closer to the neighbor robot are then reassigned to the neighbor robot (correcting the labels in the identity set $I(q)$). After the back propagation ends, the front propagation continues until the next neighbor of robot $i$ is reached (where another back propagation starts), or the algorithm terminates as the priority queue $L_F$ is empty ($M(q)$ for all nodes in the graph has been calculated) or an abort criterion is reached. Our abort criterion is based on the technique introduced in [1], where the circle around robot $i$ is iteratively enlarged to the minimum size that enables to compute the Voronoi region completely. In the discrete domain, the abort criterion is $M(l) \geq 2\, max_{(q \in Q | I(q) = I(p^*_i))} d(p^*_i, q),\ \forall l \in L_F$.

In the second step of the Lloyd algorithm, the new goal node $p^*_{i_{NEW}}$ is determined from the one-ring neighborhood of $p^*_i$. The selection of $p^*_{i_{NEW}}$ as the node the robot encounters first on the shortest path to $\gamma_{V^*_{1_i}}$ is motivated by 1) following a gradient descent procedure to minimize the cost function similar to the continuous case (see [2]) and 2) lowering the computational effort by avoiding the calculation of the intrinsic center of mass $\gamma_{V^*_{1_i}}$ of the Voronoi region $V^*_{1_i}$ itself. The new target position $p^*_{i_{NEW}}$ is the neighboring node $n_i(p^*_i)$ of $p^*_i$ which maximizes $\sum_{(q \in V^*_{1_i} | n_i(p^*_i) \in S_q)} M(q)$, where $S_q$ is the sequence of nodes in the shortest path from $p^*_i$ to $q$. Permanent decrease in the local cost $\mathcal{H}_{dis_i}(p^*_i)$ through transitions from $p^*_i$ to $p^*_{i_{NEW}} = n_i(p^*_i)$ is assured by requiring $\mathcal{H}_{dis_i}(p^*_{i_{NEW}}) < \mathcal{H}_{dis_i}(p^*_i)$.

### *B. Local cell exchange algorithm*

Our second coverage control algorithm is based on the Euclidean distance metric and minimizes the cost function (2) through local optimization steps on the mesh cells at the boundary of the Voronoi regions. The underlying structure of the algorithm, as shown in Algorithm 2, is similar to the one of the first algorithm. The robots are either in a `COMPUTE` or a `MOVE` state. In the `COMPUTE` state, robot $i$ updates its Voronoi region $V^*_{2_i}$ by exchanging boundary cells with adjacent Voronoi regions, and calculates the centroid $\gamma_{V^*_{2_i}}$. The robot then approaches this goal point during the `MOVE` state, moving along the cell centroids $\mathbf{c}_k$ on the dual graph $G'$. $G'$ is constructed from graph $G$. Each robot $i$ stores information about the mesh cells of its Voronoi region, such as the set of boundary edges and the identity of the adjacent mesh cells outside the own Voronoi region, in memory.

First all those boundary cells adjacent to free mesh cells are updated. Robot $i$ synchronizes its boundaries by requesting the identities of the boundary cells of the neighboring robots. If the free mesh cells have not been occupied by another robot since the last `COMPUTE` state, the cells are added to $V^*_{2_i}$ and its boundary is adjusted appropriately. The cells that belong to a neighbor robot and adjoin $V^*_{2_i}$ undergo the procedure of local cell exchange (see Section II). The local cell assignment that results in minimum cost is determined. In case a Voronoi region is split in two segments due to the cell exchange, the respective mesh cell will not be exchanged. Thus convergence of the algorithm is not affected; the cost is not minimized but remains unchanged. Robot $i$ updates the information about the mesh cells according to the assignment and communicates these changes to the other robots. Hence robot $i$ requests data to perform the local optimization and sends the result back to each neighbor. While this bidirectional communication takes place, both robots are not allowed to answer another request.

***Remark** 3:* The real robots move in the continuous domain. Our environment representation is a graph embedded in the surface, i.e. each node $q$ relates to a point in $\mathbb{R}^3$. Positions $\mathbf{P}$ are projected to positions $\mathbf{P}^*$ and mapped to nodes $P^*$ on the graph. In Algorithm 1, the robots' positions are represented by the next goal nodes $p^*_i = p^*_{i_{NEW}}$ during the `MOVE` state to enable calculations in the discrete domain. Algorithm 2 uses the Euclidean distance measure, i.e. the centroids $\gamma_{V^*_{2_i}}$ may fall out of the surface and need to be constrained on a triangle centroid $\mathbf{c}_k$. The closest centroid in $V^*_{2_i}$ is chosen as the new goal point, $\mathbf{p}^*_{i_{NEW}} = \mathrm{argmin}_{(\mathbf{c}_k | q' \in V^*_{2_i})} \parallel \mathbf{c}_k - \gamma_{V^*_{2_i}} \parallel$.

---

**Algorithm 2** LOCAL CELL EXCHANGE

---

**Require:** Set of robots $i \in \{1, ..., n\}$ with initial positions $\mathbf{p}^*_{i_0}$ on the surface, and each robot $i$ provided with:
- surface mesh represented by graph $G$ (known env.) or subgraph $G_i$ (unknown env.)
- localization, sensing and communication with neighbor robots (synchronous or asynchronous mode)

**loop**
  **if** (state == `COMPUTE`) **then**
    Assign free cells to Voronoi region $V^*_{2_i}$
    Compute cell assignments (local cell exchange)
    Calculate $\mathbf{p}^*_{i_{NEW}}$ ($= \gamma_{V^*_{2_i}}$ constrained on $G'$)
    Update state
  **else** $\{$state == `MOVE`$\}$
    Move to $\mathbf{p}^*_{i_{NEW}}$ and update $p^*_i$
    Update state
  **end if**
**end loop**

---

### *C. Convergence of algorithms*

In the following, graph $G$ for Algorithm 1 and its dual graph $G'$ for Algorithm 2 are assumed to be finite. We prove convergence of the algorithms in the case of known environments. The proofs for unknown environments are omitted due to limited space. They are extensions relying on the fact that the subgraphs $G_i$, or $G'_i$ respectively, covered by robots at nodes $P^*$ only change for a finite number of times given finite graphs. Convergence then results from the concatenation of sequences, in which the subgraphs remain unchanged.

***Proposition** 1 (Convergence of Algorithm 1):* A group of robots covers a graph $G$ and converges to a local minimum configuration of the cost in (2), i.e. each robot becomes a minimizer of (3), by performing Algorithm 1.

*Proof:* $D(p^*_i, q) = d(p^*_i, q)$ is strictly increasing on $S_q$, the sequence of nodes given by the shortest path from $p^*_i$ to $q$. Therefore the Voronoi partition $V^*_1$ for any fixed robot configuration $\{p^*_1, ..., p^*_n\}$ minimizes $\mathcal{H}_{dis}(P^*)$, i.e. $\mathcal{H}_{dis}(P^*, V^*_1(P^*)) \leq \mathcal{H}_{dis}(P^*, W)$ and $W$ any partition. Let $T(P^*)$ be the mapping from nodes $P^*$ to goal nodes $P^*_{NEW}$, the set of neighboring nodes $n_i(p^*_i)$ on the shortest paths to $\gamma_{V^*_{1_i}}$, $i \in \{1, ..., n\}$. In synchronous mode we have $T : \{p^*_1, ..., p^*_i, ..., p^*_n\} \longrightarrow \{p^*_{1_{NEW}}, ..., p^*_{i_{NEW}}, ..., p^*_{n_{NEW}}\}$, and in asynchronous mode $T : \{p^*_1, ..., p^*_i, ..., p^*_n\} \longrightarrow \{p^*_1, ..., p^*_{i_{NEW}}, ..., p^*_n\}$. The mapping T has the property $\mathcal{H}_{dis}(T(P^*), W) \leq \mathcal{H}_{dis}(P^*, W)$, which is guaranteed by Algorithm 1 that requires $\mathcal{H}_{dis_i}(p^*_{i_{NEW}}) < \mathcal{H}_{dis_i}(p^*_i)$ for each robot. Inequality $\mathcal{H}_{dis}(T(P^*), V^*_1(T(P^*))) \leq \mathcal{H}_{dis}(T(P^*), V^*_1(P^*))$ follows directly from the optimality of a Voronoi tessellation for a fixed set of points. Since the property of T holds for an arbitrary tessellation $W$, we finally get with $W = V^*_1(P^*)$: $\mathcal{H}_{dis}(T(P^*), V^*_1(T(P^*))) \leq \mathcal{H}_{dis}(T(P^*), V^*_1(P^*)) \leq \mathcal{H}_{dis}(P^*, V^*_1(P^*))$, and thus the cost is minimized in each iteration step of Algorithm 1. ■

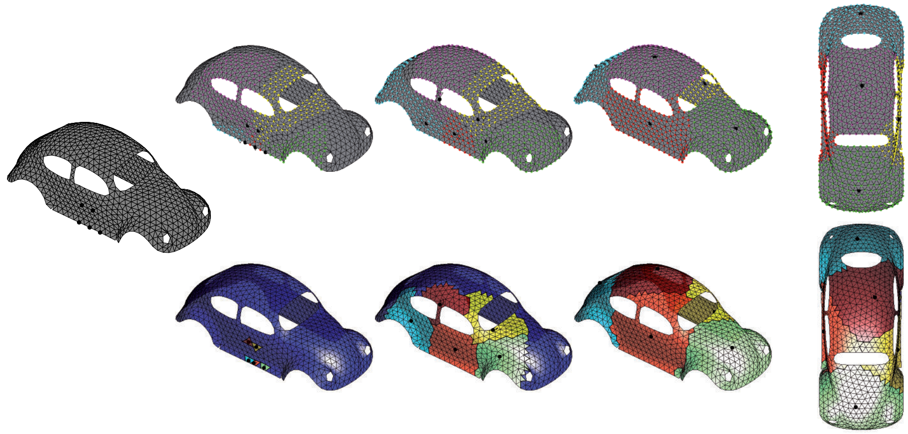


Fig. 2. Simulation on the "Beetle" 3D model. A group of five point robots covers the surface of the car using Algorithm 1 (top row) and Algorithm 2 (bottom row). Both algorithms result in full coverage of the surface mesh.

***Proposition** 2 (Convergence of Algorithm 2):* Given the connectivity of single Voronoi regions, a group of robots covers a graph $G'$ and converges to a local minimum configuration of the cost in (4) by performing Algorithm 2.

*Proof:* The Voronoi regions $V_{2_i}^*$ grow to cover the free nodes of graph $G'$. Until full coverage of the graph is reached, $\mathcal{H}_{dis\_approx}(P^*)$ increases constantly. An upper bound in the cost is given by the size of $G'$; otherwise no increase of the cost is induced. In order to maintain the connectivity of Voronoi regions, cells may not be exchanged, i.e. further minimization is suppressed but the cost is not increased either. $\mathcal{H}_{dis\_approx}(P^*)$ is positive and the exchanges of mesh cells across the boundaries of $V_{2_i}^*$ reduce the cost locally in each iteration step (in synchronous as well as asynchronous mode). A point is reached, where no cells can be exchanged to further decrease the cost. From this it follows that Algorithm 2 converges to a configuration of local minimum cost. ■

## IV. Simulations on 3D surface meshes

We use Matlab as simulation environment. The two proposed algorithms are implemented in C++ and integrated in Matlab via *.mex files. For initialization and visualization of the mesh, toolboxes from [12] are used [1] .

The two coverage control algorithms are simulated for two environments of different complexity. Figure 3 shows the first environment, a simple curved surface. A test setup was additionally built for the curved surface to realize experiments with real robots (see Section V). The second environment is a more complex 3D model, the "Beetle" model from computer graphics [1].

Figure 2 illustrates how a group of five point robots covers the surface of the car on a triangle mesh with edge lengths of 100 mm. The simulation results of Figure 2 were achieved with a sensor range $R_s$ of roughly a $1/3$ of the car length and all robots connected (infinite $R_c$) under asynchronous mode and for unknown environment. Besides, simulations for all four combinations of assumptions and with varying initial positions were run and succeeded in covering the surface. The simulation for the "Beetle" 3D model demonstrates the basic functionality of our approach to cover arbitrary surfaces in 3D space.

The behavior of the two algorithms is further analyzed for an increasing number of robots. 20 simulation runs were executed per group, with groups of 5, 10 and 20 robots, on a 50 mm mesh of the curved surface, in synchronous mode and known environment. The initial configuration of the robots on the mesh was selected at random for each run and thus the simulations converged to different local minima. The graphs with the plots of all 20 simulations (black lines) with their final values (blue squares) are visualized for the group with 10 robots for both algorithms in Figure 4. The average convergence time is 6.5 s for Algorithm 1 and 24.8 s for Algorithm 2 (red dashed line) and the average final cost (blue dashed line) for Algorithm 1 is 18.8 % and for Algorithm 2 6.5 % above the best possible final cost (black dashed line). The best possible final cost approximates the global minimum of $\mathcal{H}_{dis}$ and is computed as an upper bound by the Lagrangean relaxation heuristics according to [19]. The best possible cost at final configuration decreases due to the summation over smaller and smaller distances with increasing number of robots (for 5 robots: $2.23 \cdot 10^7$, 10 robots: $1.45 \cdot 10^7$, 20 robots: $0.52 \cdot 10^7$). On the other hand, we found that the average deviation in cost increases approximately linear in the number of robots. A reason for this effect is the increasing robot density on the graph, which causes the robots to block each other when badly initialized.

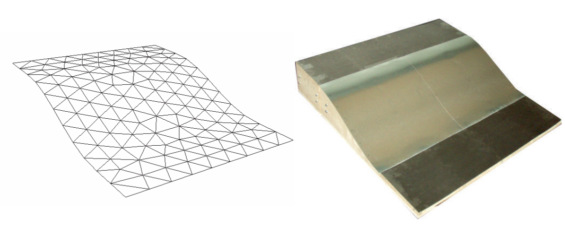
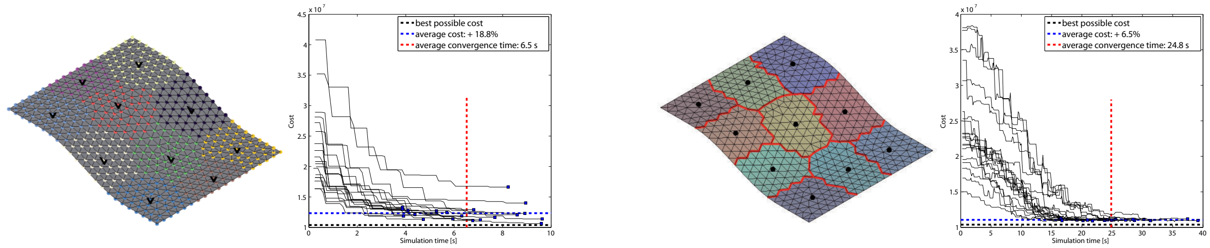


Fig. 3. Test environment: triangle mesh and real setup of curved surface.

[1]The reader refers to *www.ceremade.dauphine.fr/∼peyre*




(a) (b)

Fig. 4. Simulation on the curved surface. A group of ten robots covers the surface mesh. The best final configurations and the cost from 20 simulation runs are shown for (a) Algorithm 1 and (b) Algorithm 2. The best possible cost, average final cost and average convergence time are added in the plots.

***Remark** 4:* As the cost in (4) is approximated and the Voronoi regions grow during execution of Algorithm 2, the developing of the cost functions of Algorithm 1 and Algorithm 2 cannot be compared directly. Therefore the cost of Algorithm 2 is recalculated for evaluation purposes based on the latest robot positions in each time step by using Equation (2) as applied in Algorithm 1.

## V. Experiments on Real Surface

The test environment in Figure 3 is built from plywood and a thin carbon steel sheet covered with a flexible magnet is laid on top of the wooden frame. The base area measures 1.2 x 1.2 $m^2$. Five e-puck robots [20] are used for the experiments. They come with a bluetooth module, a Li-ion battery and two differentially-driven wheels, and have additionally been equipped with markers on the top for tracking and small magnets at the bottom for climbing ferromagnetic surfaces. We kept the kinematic model of the robots simple: the e-pucks rotate until their front direction is near to parallel to the goal direction (within $\pm$ 5°) and then drive straight to the goal.

The e-pucks receive the commands via bluetooth. An overhead camera (sampling rate of 0.5 - 1 Hz, resolution of 1280 x 1024 pixels, maximum distance to surface of 1 m) captures the frames and passes them on to the base station and further to Matlab. The image is read in the ARToolkit [21], which extracts the markers and delivers the position and orientation of each robot (the final position accuracy was evaluated to be a ball of radius 3 cm around the robots' centers). Depending on the state of the robot, the next goal position is calculated by Algorithm 1 or Algorithm 2 (`COMPUTE` state), or the new control inputs are sent via bluetooth to the e-pucks (`MOVE` state), which closes the control loop. The memory as well as the data transfer between the robots are simulated in Matlab. However, all the calculations are executed decentralized and the robots form a distributed system.

The performance of Algorithm 1 and Algorithm 2 was analyzed with the five robots and the mesh from Figure 3 with edge lengths of 50 mm, 100 mm and 200 mm. The 200 mm mesh is too coarse for Algorithm 1 and the robots get stuck very easily in a suboptimal local minimum. Algorithm 2 is more robust against the size of the mesh cells and also works for 200 mm meshes. Basically, coarser meshes have the advantage of faster convergence as the number of nodes and mesh cells, and thus the computational cost, are greatly reduced. Additionally, the robots move over longer distances until they reach a next `COMPUTE` state, where they rotate before driving straight again. The reduction in rotations saves additional time. Convergence time grows by a factor of 1.5 at each transition from the 200 mm mesh to the 100 mm mesh and the 100 mm mesh to the 50 mm mesh. In conclusion, the 100 mm mesh showed fast convergence to a good final tessellation in simulation as well as in experiments. The number of nodes is ideally as low as possible, but without losing details of the structure due to low mesh resolution.

In general, the robots converge faster to the final configuration for experiments in asynchronous mode than in synchronous mode. As expected, the robots do not have to wait for the other robots when computing their next goal positions, which speeds up the distribution. The time difference is more distinct for Algorithm 2 since the robots can drive along several triangle centroids until they reach their goal point, i.e. a sequence in the `MOVE` state may take longer than in Algorithm 1. This results in longer waiting times in the synchronous mode.

In total over 50 experiments were run for each of the two algorithms on the test environment in Figure 3. Thereof 25 experimental runs were performed in asynchronous mode and for unknown environment, which are the settings that may be most desirable in a real application. A triangle mesh with triangle edges of 100 mm was used. The sensor range $R_s$ was set to 0.5 m and the communication range $R_c$ was kept infinite. A sequence from an experimental run is shown in Figure 6 for both of the algorithms. The video accompanying this paper [2] offers further illustration of the coverage of curved surfaces. The five robots' initial configuration was chosen as shown in the top of Figure 6(a) and 6(b).

Over the 25 experimental runs, Algorithm 1 results in an average convergence time of 137 s with a standard deviation of 28.6 s (see Figure 7(a) on the right). Algorithm 2 needs longer to converge than Algorithm 1 and evaluations in Figure 7(b) show an average convergence time of 303 s with standard deviation of 55.8 s. The lower convergence rate

[2]The reader also refers to *www.asl.ethz.ch*

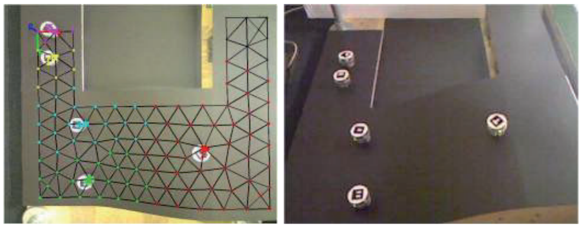


Fig. 5. Suboptimal distribution of robots on the curved surface with hole. Under Algorithm 1 the five e-pucks get stuck in a suboptimal local minimum when starting from the initial configuration depicted in Figure 1 (left: top view with surface mesh and Voronoi partition overlay, right: side view).

is in accordance with the simulation results and is caused by “zig-zag movement” and “cell growing” of Algorithm 2. The robots move from triangle centroid to triangle centroid on straight lines. These trajectories describe a zig-zag path and the robots rotate more often than for Algorithm 1. In addition, the Voronoi regions are growing cell by cell and thus are created slower than with Algorithm 1.

Algorithm 1 mostly converges to the same final configuration over 25 experimental runs in asynchronous mode and unknown environment (see Figure 7(a) on the left). The final configuration of Algorithm 1 strongly depends on the initial configuration. Experiments with Algorithm 2 show that the robots distribute into different final configurations (see Figure 7(b) on the left). The order, in which the boundaries of the Voronoi regions are updated, influences the cell assignments and the Voronoi regions evolve differently from run to run. Thereby Algorithm 2 reaches slightly lower values in cost compared to Algorithm 1. The average cost of the final configuration for Algorithm 2 lies 10.4 % above the best possible cost, whereas the average cost of Algorithm 1 is 12.8 % above. Besides, the final cost of Algorithm 2 varies in a bigger range since different minima of the cost are reached in the final configurations.

For our last experiment, we modified the test environment by removing the middle part from the test setup to add a hole obstacle to the curved surface. Coverage control now must deal with an additional non-convexity. The resulting coverage is shown in Figure 1 for Algorithm 2 and Figure 5 for Algorithm 1 on a 100 mm mesh in asynchronous mode and unknown environment. While Algorithm 2 generates well-balanced partitions, Algorithm 1 converges to a local minimum with suboptimal distribution of the robots on the surface (see Figure 5). A reason for the suboptimal coverage is the strong dependence of Algorithm 1 on the initial configuration and the unfavorable start positions in this particular case (all the robots are lined up one after the other on the narrow branch). Tests with the 50 mm mesh showed that the use of a denser mesh improves the final partitions, however enhanced methods on the side of the algorithms, such as equitable partitioning through different edge weights, must be investigated.

## VI. CONCLUSIONS

We have introduced an approach to deploy a group of robots on a non-planar surface in 3D space in this paper. Two coverage control algorithms have been presented that achieve discrete Voronoi coverage on a surface mesh. The front propagation algorithm uses label correcting methods and the Lloyd algorithm to directly cover the graph. The local cell exchange algorithm approximates the Voronoi regions by mesh cells and locally reassigns cells to adjacent regions to minimize the cost of the overall Voronoi configuration. Both algorithms have been tested thoroughly. Simulations on a complex 3D model and with varying number of robots verified the functionality of the algorithms. Experiments with a group of five e-puck robots were conducted on a test setup with curved surface and showed convergence to good final configurations. The analysis further revealed some complementary trends in the characteristics of the two coverage control algorithms: the convergence rate is higher for Algorithm 1 than Algorithm 2, Algorithm 1 is more likely to converge to a suboptimal local minimum and the cost of the final configuration remains above the cost of Algorithm 2, the initial configuration influences the performance of Algorithm 1 more than for Algorithm 2 as observed from tests on the curved surface with hole, the deterioration of partitions due to defects or constraints in the mesh is generally lower for Algorithm 1 compared to Algorithm 2, and the requirements for computation and communication among the robots differ for the two algorithms.

In our future work, we are going to extend the algorithms to account for varying curvature and to adapt the final tessellation to edges and obstacles on the surface. Future approaches could also search for improvements by combining elements of the two algorithms. A further direction in our reseach is to embed the algorithms on robots with higher climbing mobility and robot-to-robot communication, also taking uncertainties into account. Eventually, as both control algorithms are based on surface meshes, we will investigate ways to generate 3D meshes from the environment directly on the robots.

## REFERENCES


[1] J. Cortés, S. Martinez, T. Karatas and F. Bullo, ”Coverage Control for Mobile Sensing Networks”, *IEEE Trans. on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

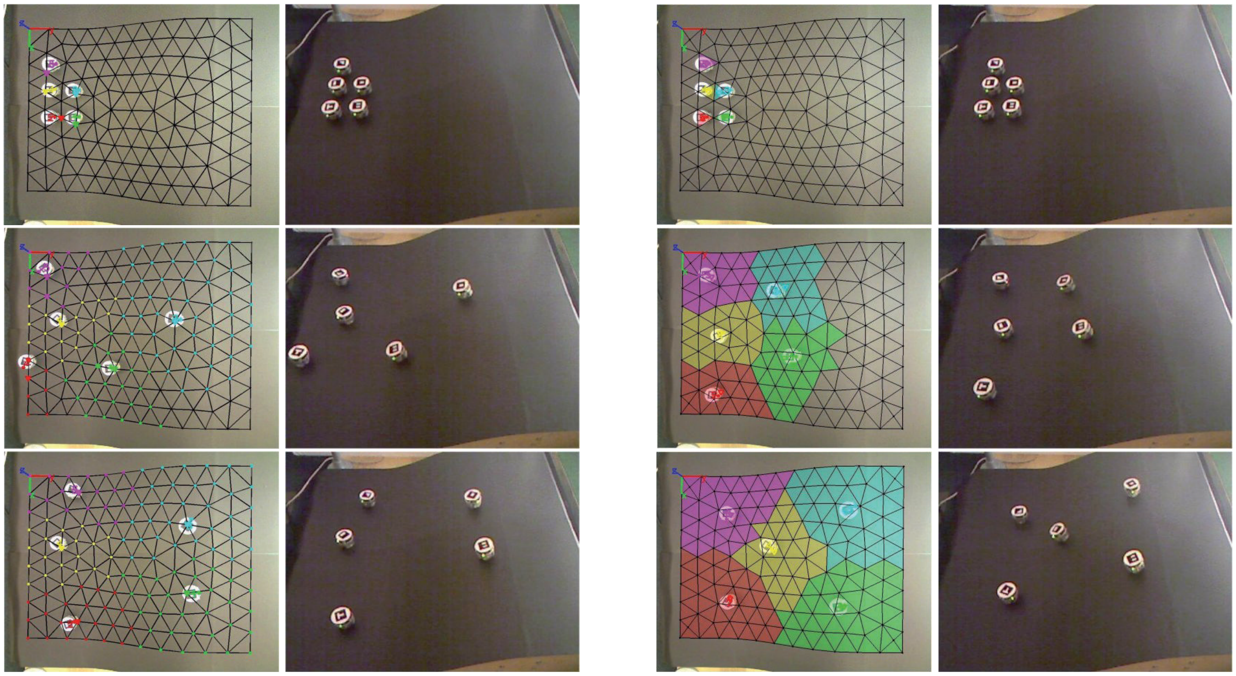
[2] L. Pimenta, V. Kumar, R. Mesquita and G. Pereira, ”Sensing and Coverage for a Network of Heterogeneous Robots”, in *Proc. of the 47th IEEE Conference on Decision and Control*, 2008.

[3] M. Schwager, D. Rus and J.-J. E. Slotine, ”Decentralized, Adaptive Coverage Control for Networked Robots”, in *I. J. Robotic Res.*, vol. 28, no. 3, pp. 357–375, 2009.

[4] M. Schwager, B. Julian and D. Rus, ”Optimal Coverage for Multiple Hovering Robots with Downward-Facing Cameras”, in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, pp. 3515–3522, 2009.

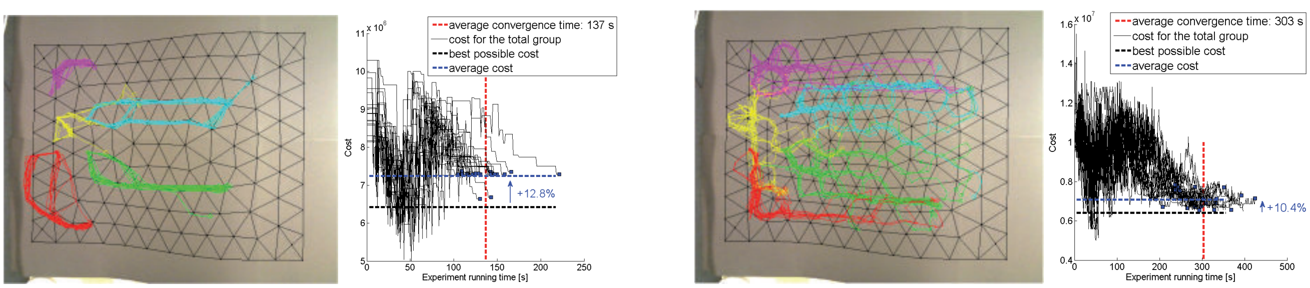
[5] Q. Du, M. Gunzburger and L. Ju, ”Constrained Centroidal Voronoi Tessellations for Surfaces”, *SIAM J. Sci. Comput.*, vol. 24, no. 5, pp. 1488–1506, 2002.

[6] M. Zhong and C. G. Cassandras, ”Distributed Coverage Control in Sensor Network Environments with Polygonal Obstacles”, in *IFAC World Congress*, pp. 4162–4167, 2008.



(a) (b)

Fig. 6. Experiment on curved surface: (a) Algorithm 1 and (b) Algorithm 2 for the group of five e-pucks in asynchronous mode and unknown environment. The images from the overhead camera in the left columns show the initial, intermediate and final configurations during coverage of the surface (the image is overlayed with the surface mesh and the Voronoi partitions). The nodes/cells assigned to a robot are visualized in the color of the robot. The images from a second camera (mounted at 45° on the side) in the right columns show the distribution of the robots on the surface for the same timestamps.




(a) (b)

Fig. 7. Statistical evaluation of 25 experimental runs on the curved surface: (a) Algorithm 1 and (b) Algorithm 2 for the group of five e-pucks in asynchronous mode and unknown environment. The trajectories of the robots during deployment on the surface mesh are shown on the left. The cost for the total group is plotted over running time on the right. The best possible cost, average final cost and average convergence time are added in the plots.

[7] C. Caicedo and M. Žefran, "A coverage algorithm for a class of non-convex regions", in *Proc. of the 47th IEEE Conference on Decision and Control*, pp. 4244–4249, 2008.

[8] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots", in *Proc. of the International Conference on Robotics and Automation (ICRA 10)*, 2010.

[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Mesh optimization", in *Proc. of ACM SIGGRAPH*, pp. 19–26, 1993.

[10] N. Amenta, S. Choi and R. K. Kolluri, "The Power Crust", in *6th ACM Symp. on Solid Modeling and Applications*, pp. 249–260, 2001.

[11] P. Alliez, É. C. de Verdière, O. Devillers and M. Isenburg, "Centroidal Voronoi diagrams for isotropic surface remeshing", *Graph. Models*, vol. 67, no. 3, pp. 204–231, 2005.

[12] G. Peyré and L. Cohen, "Surface Segmentation Using Geodesic Centroidal Tesselation", in *Proc. of the 2nd Int. Symp. on 3D Data Processing, Visualization and Transmission*, pp. 995–1002, 2004.

[13] S. Valette and J.-M. Chassery, "Approximated Centroidal Voronoi Diagrams for Uniform Polygonal Mesh Coarsening", *Computer Graphics Forum*, vol. 23, no. 9, pp. 381–389, 2004.

[14] B. Yamauchi, "Frontier-based exploration using multiple robots", in *Proc. of the 2nd International Conference on Autonomous Agents (AGENTS 98)*, pp. 47–53, 1998.

[15] J. W. Durham, R. Carli, P. Frasca and F. Bullo, "Discrete Partitioning and Coverage Control with Gossip Communication", in *ASME Dynamic Systems and Control Conference*, 2009.

[16] S. Lloyd, "Least squares quantization in PCM", *IEEE Trans. on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[17] M. Erwig, "The Graph Voronoi Diagram with Applications", *Networks*, vol. 36, pp. 156–163, 2000.

[18] D. P. Bertsekas, "Dynamic Programming and Optimal Control", *Athena Scientific*, vol. 1, 3rd edition, 2005.

[19] J. E. Beasley, "Lagrangean heuristics for location problems", *European journal of operational research*, 1993.

[20] F. Mondada et al., "The e-puck, a Robot Designed for Education in Engineering", in *Proc. of the 9th Conference on Autonomous Robot Systems and Competitions*, pp. 59–65, 2009.

[21] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System", in *Proc. of the 2nd International Workshop on Augmented Reality*, 1999.