Voronoi Coverage of Non-Convex Environments with a Group of Networked Robots

2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Voronoi coverage of non-convex environments with a group of networked robots

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

***Abstract*— This paper presents a solution to decentralized Voronoi coverage in non-convex polygonal environments. We show that complications arise when existing approaches to Voronoi coverage are applied for deploying a group of robots in non-convex environments. We present an algorithm that is guaranteed to converge to a local optimum. Our algorithm combines classical Voronoi coverage with the Lloyd algorithm and the local path planning algorithm TangentBug to compute the motion of the robots around obstacles and corners. We present the algorithm and prove convergence and optimality. We also discuss experimental results from an implementation with five robots.**

I. Introduction

Distributed coverage is a common application for multi-robot systems where robots spread over an environment to fulfill a perception task. Examples include environmental monitoring and surveillance, maintenance and inspection, or distributed actuation tasks, such as harvesting or demining. An early survey about multi-robot coverage by [1] introduces three basic types of coverage problems: blanket, sweep and barrier coverage. Blanket coverage methods deploy sensors, e.g. carried by networked robots, in a static arrangement to cover an area. Most current results on blanket coverage are focused on convex environments; however realistic applications require the ability of coping with non-convex environments, including areas with many free-standing obstacles as well as areas without obstacles but with non-convex boundaries. To pave the way toward real-world multi-robot systems, it is essential to enable coverage by groups of robots in such environments. We address the deployment of a group of networked robots to cover a non-convex environment in this paper.

We propose a new control strategy that combines two well-known algorithms: *Lloyd algorithm* [11], originated from quantization theory, and *TangentBug* [14], a derivative of the family of Bug algorithms. Figure 1 illustrates the concept of our approach: Lloyd algorithm updates the goal position of each robot, TangentBug then plans a path to the given goal. The consecutive interaction between the two algorithms results in global convergence of the robot team to a final Voronoi configuration that optimizes a cost function. The control strategy spreads the robots over the environment while taking care of the non-convexity of the environment by the built-in obstacle avoidance behavior.

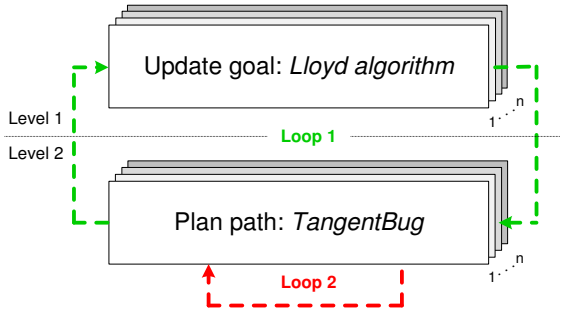

Fig. 1. Control strategy as combination of the Lloyd algorithm and the TangentBug algorithm. Hierarchical approach running distributed on n robots: Lloyd algorithm computes the Voronoi region and updates the current target (*level 1*), the local path planner computes the path to the target and performs obstacle avoidance (*level 2*). A first control loop (*loop 1*) is formed between level 1 and level 2, a second control loop (*loop 2*) is built into the path planner on level 2.

As our control strategy combines multi-robot coverage and robot navigation, there is a broad body of work with potential influence. Artificial potential fields, vector field histograms and the Bug algorithms are examples of simple but powerful sensor-based local path planners. The actual coverage control in our control strategy is mainly based on a well-known method for coverage of convex environments with mobile networked robots. Voronoi coverage was introduced in [2] and further developed in [3], [4] among others. The basic concept goes back to the continuous p-median problem, which is part of the locational optimization framework [5]. We will refer to this control stategy as *Voronoi coverage* in this paper.

Other works have investigated the control of multiple robots in non-convex environments. [6] considers the coordination of a group of robots to achieve rendez-vous in a non-convex environment. [7] focuses on coverage to solve the distributed Art Gallery Problem in non-convex environments. A control approach to drive a multi-robot team to target sets under collision avoidance and maintaining proximity constraints is presented in [8] for known environments with obstacles. In [9] and [10] the environment is transformed by a diffeomorphism to a corresponding convex region, in

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, jc_metzger@student.ethz.ch, rsiegwart@ethz.ch

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

which regular Voronoi coverage can be applied. The method offers a creative solution to non-convex coverage, however the authors note that it presents significant computational challenges, and that it may lead to solutions that differ from the optimal coverage solutions in the original space. In [10] the authors resolve some of these issues for convex regions with isolated obstacles. [4] approaches coverage of environments with non-convex boundaries by applying the geodesic distance measure to Voronoi coverage. This method provides an elegant solution for some classes of environments but is not guaranteed to work for all types of non-convex environments.

This paper is organized as follows. In Section II we present basic concepts of the Voronoi coverage method for convex environments. We point out complications of the method in non-convex environments and introduce our new approach to the problem. In Section III we describe the algorithm and its implementation in detail. We prove convergence and optimality of our control strategy in Section IV. Simulations in Section V and physical experiments in Section VI with a group of networked robots in a non-convex environment further verify the concept. We conclude in Section VII and point to some interesting extensions of our approach.

II. Problem formulation

A group of networked robots must be deployed in an environment. The environment can either be convex (forming a convex domain without any obstacles or holes in it) or it can be non-convex (including free-standing obstacles or holes and areas with non-convex boundaries). Our focus is on non-convex environments, but our approach also applies for convex environments.

A. Convex environments

A widely studied class of solutions to coverage in convex environments uses Voronoi coverage to optimize the configuration of n robots by minimizing the average distance to the nearest points of interest. A team of n robots at positions $\mathbf{P} = [\mathbf{p}_i]_{i=1}^n \in \mathbb{R}^{Nn}$, navigate in a bounded polygonal environment $\Omega \subset \mathbb{R}^N$. Ω is a closed set with the boundary $\partial\Omega$. For the Euclidean distance measure with $\|\cdot\|$ denoting the ℓ^2- norm, a *Voronoi tessellation* $V = \{V_i\}_{i=1}^n$ is a partition of Ω for which

$$V_i = \left\{ \mathbf{x} \in \Omega \,|\, \|\mathbf{x}-\mathbf{p}_i\| \leq \|\mathbf{x}-\mathbf{p}_j\|, \forall j = 1,...,n\,,\, j \neq i \right\}. \quad (1)$$

Each V_i is the Voronoi cell corresponding to a robot located at $\mathbf{p}_i$. The positions $\mathbf{p}_i$ are called the generating points, the robots are the generators of the Voronoi tessellation. Given a positive density function $\phi : \Omega \to \mathbb{R}_{>0}$, we can define the mass and the centroid of V_i as

$$M_{V_i} = \int_{V_i} \phi(\mathbf{x})\, d\mathbf{x}, \quad \mathbf{C}_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} \mathbf{x}\ \phi(\mathbf{x})\, d\mathbf{x}\,. \quad (2)$$

A Voronoi tessellation becomes a centroidal Voronoi tessellation (CVT) when the generating points $\mathbf{p}_i$ coincide with the centroids, i.e. $\mathbf{p}_i = \mathbf{C}_{V_i}, \forall i \in \{1,...,n\}$. In this case we say the robots reach a *centroidal Voronoi configuration*. CVTs are known to locally minimize the cost function

$$\mathscr{H}(\mathbf{P}) = \sum_{i=1}^n \mathscr{H}(\mathbf{p}_i) = \sum_{i=1}^n \int_{V_i} f(D(\mathbf{p}_i, \mathbf{x}))\ \phi(\mathbf{x})\, d\mathbf{x}\,, \quad (3)$$

with $D(\cdot)$ a distance measure between a point $\mathbf{p}_i$ and a location $\mathbf{x}$ in Ω. The density function $\phi(\cdot)$ describes the weighting or importance of different areas in Ω. The function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is assumed to be smooth and strictly increasing. $f(z) = \frac{1}{2}z^2$ is our choice in the remainder of the paper for simplicity. It accounts for the sensor reliability of a robot and adds to the cost as the distance from the sensor increases.

1) Euclidean distance: The Euclidean distance $D(\mathbf{p}_i, \mathbf{x}) = \|\mathbf{x}-\mathbf{p}_i\|$ is a standardized choice for open space and convex environments. Using the Euclidean distance we get from (3)

$$\frac{\partial \mathscr{H}(\mathbf{p}_i)}{\partial \mathbf{p}_i} = -M_{V_i}(\mathbf{C}_{V_i} - \mathbf{p}_i) = \mathbf{0}, \quad (4)$$

where M_{V_i} and $\mathbf{C}_{V_i}$ were defined in (2). The local minima are reached with a CVT, since $\mathbf{C}_{V_i} = \mathbf{p}_i$ implies $\frac{\partial \mathscr{H}}{\partial \mathbf{p}_i} = \mathbf{0},\ \forall i \in \{1,...,n\}$.

2) Geodesic distance: The geodesic distance measure is another intuitive choice, especially in the context of non-convexity [4]. The geodesic distance is the length of the shortest path from a start point $\mathbf{p}_i$ to a goal point $\mathbf{x}$, so that the path lies completely in the domain Ω.

A deterministic method to construct centroidal Voronoi tessellations is the classic Lloyd algorithm [11]: 1) construct the Voronoi partition for the generating points at $[\mathbf{p}_i]_{i=1}^n$, 2) compute the centroids of these Voronoi regions, 3) set the new locations of $[\mathbf{p}_i]_{i=1}^n$ to the centroids and start all over again. Voronoi coverage is based on the Lloyd algorithm and its continuous and discrete-time versions.
From (4) the proportional control law (here shown for the Euclidean distance measure)

$$u_i = \dot{\mathbf{p}}_i = -\frac{k}{M_{V_i}} \frac{\partial \mathscr{H}(\mathbf{p}_i)}{\partial \mathbf{p}_i} = k\,(\mathbf{C}_{V_i} - \mathbf{p}_i)\,, \quad (5)$$

with simple first-order dynamics $\dot{\mathbf{p}}_i = \mathbf{u}_i$ can directly be derived. From (5) it becomes clear that the Lloyd algorithm implements a gradient descent controller. The convergence properties of the Lloyd algorithm are well established. We repeat them here as it will be useful later in our analysis.

***Lemma** 1 (Convergence of Lloyd algorithm):* A set of points in a given environment converges under the Lloyd algorithm to a centroidal Voronoi configuration.

Proof: Convergence follows from Prop. III.3 in [2]. ■

B. Complications from non-convex environments

We demonstrate the possible difficulties that arise for the coverage problem in a non-convex environment with the different example configurations shown in Figure 2(a) – (d). For clarity of our explanation, we base our analysis on a single robot in a non-convex environment. But the considerations also hold true for the multi-robot case, where

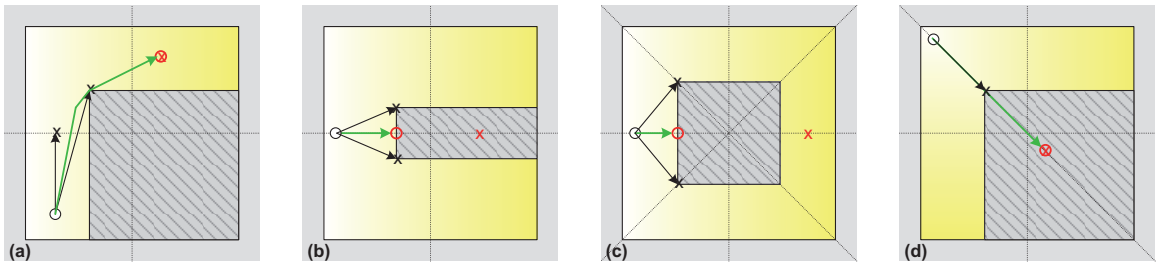

Fig. 2. Non-convex example environments (white circle: robot initial position, red circle: robot final position (geodesic), red cross: target (Euclidean, CVT), green line: robot path, yellow color gradient: density function $\phi(\cdot)$). (a) L-shaped region: the geodesic distance provides a reasonable robot path (see also [4]). (b) U-shaped region: the robot converges to a minimum point arbitrarily far from the optimal goal point (in the obstacle) due to the compromise character of the geodesic distance. (c) Region with free-standing obstacle: the robot get stuck on its way to the goal in a saddle point as it does not stop making compromises between different shortest path options. (d) L-shaped region (with different robot start position and different $\phi(\cdot)$ compared to (a)): Even in the case of a L-shaped region, when the start position and thus the robot's visibility change, the robot happens to leave the region because of non-zero gradient sums.

a Voronoi region of a single robot takes shape of the regions in the example configurations.

First consider applying a controller (5) as used for Voronoi coverage in convex environments. Let robot i at position $\mathbf{p}_i$ drive on straight lines, according to the Euclidean distance measure, to the target position $\mathbf{t}_i$, which is the centroid $\mathbf{C}_{V_i}$ of its Voronoi cell V_i. We are confronted with two types of critical configurations: (i) the robot position temporarily leaves the environment Ω during motion, i.e. the path goes through an obstacle (Figure 2(a), (c)), (ii) the final goal position lies outside the environment, in an obstacle, and cannot be reached (Figure 2(b), (d)).

Next we investigate the limitations of Voronoi coverage in non-convex environments when using the geodesic distance measure. Although the geodesic distance helps keeping the robot inside the environment region on its way to the target and works fine for some cases, e.g. for the example in Figure 2(a), there are other cases where it does not (such as the cases depicted in Figure 2(b) – (d)). The geodesic distance measure calculates the paths along the boundaries and avoids the obstacles. But it introduces a *compromise characteristic* at the same time, i.e. the controller gets trapped and remains at a position of minimum average distance to the locations of interest (see Figure 2(b)). This may entirely be in the sense of locational optimization as the distances to the upper and lower branch become balanced for the density distribution given in Figure 2(b). However, the compromise can lead to arbitrarily suboptimal configurations for applications that require the robot's final position to be close to the centroid and maximum density area (e.g. if the robots can see the area beyond the obstacles or sense a signal through the walls).

The compromise character further causes the robot to get stuck in a saddle point (unstable generalized CVT) in Figure 2(c) or even worse, to drive into an obstacle in Figure 2(d), as the projections of the gradients into target direction of the geodesic distance do not add up to zero when reaching the boundary. Geodesic distance as distance measure on its own solves some of the problems, but provides no general guarantee that the robot reaches a point close to a local minimum or remains in the environment. To break the trade-offs, to circumnavigate the obstacles and solve the problems pointed out in Figure 2(b) – (d), an obstacle avoidance behavior or a *local path planner* is required.

That means with respect to Voronoi partitioning, we can formulate the constrained optimization problem

$$\min_{\mathbf{P}} \mathscr{H}(\mathbf{P}) = \min_{[\mathbf{p}_i]_{i=1}^n \in \Omega} \int_{V_i} f(D(\mathbf{p}_i, \mathbf{x}))\, \phi(\mathbf{x})\, d\mathbf{x}\,, \qquad (6)$$

with Ω the constraint set. The Voronoi tessellation becomes a constrained centroidal Voronoi tessellation (CCVT) and the centroid a constrained centroid in this case (see also [12]). The robots form a *constrained centroidal Voronoi configuration*.

C. A new approach for non-convex environments

Non-convex domains pose non-convex optimization problems with non-convex constraints. We present a new approach for Voronoi coverage of a non-convex environment that builds on the Lloyd algorithm and TangentBug [14], a local path planner with obstacle avoidance behavior. The control strategy is composed of two layers of abstraction: (1) Lloyd algorithm provides goal updates based on successive computation of Voronoi regions and their centroids on the upper layer (*level 1*), while (2) TangentBug plans the robot path to the next centroid target position on the lower layer (*level 2*). This can be formulated according to Figure 1 as a continuing sequence of the two loops on level 1 and level 2, executed in a distributed fashion on each of the robots.

TangentBug is a simple but efficient sensor-based planner, capable of handling unknown environments by using a range sensor. The range can be any value from zero (contact sensor) to infinity (entire visibility domain), where the length of the robot's path usually decreases with increasing range of the sensor. TangentBug shows the two characteristic Bug behaviors: "motion-toward-target", a form of gradient descent, and "boundary-following", a form of exploration of the obstacle boundary, which both are desired behaviors in the context of our control strategy and guarantee global convergence to a target.

We next restate the convergence of TangentBug in following lemma.

***Lemma** 2 (Convergence of TangentBug):* The TangentBug algorithm converges globally toward a reachable target inside a given planar environment for a sensor of any range in a finite path.

Proof: Convergence directly follows from Theorem 1 and Theorem 2 in [13] for a contact sensor. The proof in case of a non-zero range sensor follows the lines of the proofs for the contact sensor, and it can similarly be shown that the robot reaches the target in a finite path if target reachability is given (see Theorem 1 and 2 in [14]).

■

III. VORONOI COVERAGE WITH LOCAL PATH PLANNING

We describe the proposed control strategy in detail in this Section and provide an implementation that enables present Voronoi coverage algorithms to deal with non-convex environments. A description of the implemented navigation algorithm, or non-convex coverage algorithm respectively, is detailed in Algorithm 1. Algorithm 1 calls Algorithm 2, the path planning algorithm, as subroutine. Each algorithm implements one of the two loops 1 and 2 of the control strategy, as outlined in Figure 1. We used the Lloyd algorithm on level 1 to generate the Voronoi tessellation. The Euclidean distance is used to implement the control law in this paper (equation (5)), even though the geodesic distance measure could just as well be applied.

Algorithm 2 is intentionally kept in a rather theoretic description, for the purpose of generality and clarity, and needs for well engineered implementations. We have chosen TangentBug as planner for the path generation on level 2. Although TangentBug only needs local knowledge of the obstacles, global knowledge of the environment is assumed for the execution of the Lloyd algorithm in our present approach.

A. Non-convex coverage algorithm

For the algorithm descriptions we need to introduce some new terminology. We follow a similar idea as in [15], and introduce virtual generators to navigation and path planning. We distinguish between *real generators* $\mathbf{g}_i^{real}$ and *virtual generators* $\mathbf{g}_i^{virt}$ as well as *real targets* $\mathbf{t}_i^{real}$ and *virtual targets* $\mathbf{t}_i^{virt}$ of robot i. $\mathbf{g}_i^{real}$ stands for the actual robot position, whereas $\mathbf{g}_i^{virt}$ is the desired robot position in disregard of the obstacles in the environment, as if we were dealing with a convex environment. $\mathbf{t}_i^{virt}$ is the centroid of the current Voronoi region, which was computed based on $\mathbf{g}_i^{virt}$ at the last update of loop 1. $\mathbf{t}_i^{real}$ is the projected point $\mathbf{p}_i^*$ of $\mathbf{t}_i^{virt}$ to the environment Ω. In some situations, the virtual and real points simply coincide. An *obstacle boundary Voronoi region* is defined as the subset of all the Voronoi regions, for which the condition $V_i \cap \partial\Omega \neq \emptyset \ \wedge \ \mathbf{t}_i^{virt} \notin \Omega$ applies. It is part of the constrained minimization problem and is used as condition to determine if the boundary constraint is active in a region V_i.

The proposed control strategy computes the Lloyd algorithm using only the virtual generators, which are able to freely pass through obstacles and occlusions. The robots then attempt to follow the virtual generators' centroids. Throughout execution of loop 2, virtual generators are moving toward virtual targets ($\mathbf{g}_i^{virt} \rightarrow \mathbf{t}_i^{virt}$) in a simulated environment representation where obstacles and non-convex segments of the boundary do not exist. In parallel, real generators are moving toward real targets ($\mathbf{g}_i^{real} \rightarrow \mathbf{t}_i^{real}$), while the robots approach the real targets in the real environment taking obstacles into account. Each robot computes the next virtual and real target points upon arrival at the current real target point. Each robot pretends to be on track for an ideal convex case and communicates its simulated virtual generator position to the neighbors. That leads to a situation where the robots update their own Voronoi region, centroid and thus next target point based on the virtual positions $\mathbf{g}_i^{virt}$ of their neighbors, while each of the robots tries to get as close as possible to

Algorithm 1 NON-CONVEX COVERAGE ALGORITHM

Require: Set of robots $i = \{1, ..., n\}$ with initial positions $\mathbf{p}_i^S$ in environment Ω, and each robot i provided with:
- Localization and knowledge of $\phi(\cdot)$ and Ω
- Voronoi region computation
- PATH PLANNING ALGORITHM[1]

1: initialize at time $T_i^S = 0$: $\mathbf{g}_i^{real} \leftarrow \mathbf{p}_i^S$, $\mathbf{g}_i^{virt} \leftarrow \mathbf{p}_i^S$
2: **loop** {Loop 1}
3: acquire positions $\mathbf{p}_i$ and $\{\mathbf{g}_j^{virt}\}_{j=1}^k$, $j \neq i$ of k neighbors
4: construct local Voronoi region V_i associated with $\mathbf{g}_i^{virt}$
5: compute the mass centroid $\mathbf{C}_{V_i}$ of the Voronoi region,
⇒ update *virtual target* position: $\mathbf{t}_i^{virt} \leftarrow \mathbf{C}_{V_i}$
6: run PATH PLANNING ALGORITHM
7: **end loop**
8: compute the final Voronoi region associated with $\mathbf{g}_i^{real}$

[1] PATH PLANNING ALGORITHM is a convergent, standard navigation algorithm with local path planning capability. In our case, the *TangentBug* algorithm is used.

Algorithm 2 PATH PLANNING ALGORITHM

Require: Set of robots $i = \{1, ..., n\}$ in environment Ω, and each robot i provided with:
- Obstacle avoidance: sensing and computation
- Virtual target $\mathbf{t}_i^{virt}$, and $\mathbf{var} \leftarrow \mathbf{t}_i^{virt}$

1: **loop** {Loop 2}
2: **if** V_i is an obstacle boundary Voronoi region **then**
3: project $\mathbf{t}_i^{virt}$ to point $\mathbf{p}_i^*$ onto $\partial\Omega$, and set $\mathbf{var} \leftarrow \mathbf{p}_i^*$
4: **end if**
⇒ update *real target* position: $\mathbf{t}_i^{real} \leftarrow \mathbf{var}$
5: execute next motion step toward real target $\mathbf{t}_i^{real}$, apply obstacle avoidance to drive to next position $\mathbf{p}_i$
⇒ update *real generator* position: $\mathbf{g}_i^{real} \leftarrow \mathbf{p}_i$
6: simulate next motion step toward virtual target $\mathbf{t}_i^{virt}$
⇒ update *virtual generator* position $\mathbf{g}_i^{virt}$
7: **end loop**
8: **return** virtual generator $\mathbf{g}_i^{virt}$

the set ideal target $\mathbf{t}_i^{virt}$ to reach its objective. The virtual generators and targets allow for the implementation of the Lloyd algorithm for non-convexity and maintain convergence of the method. If the robots' real positions $\mathbf{g}_i^{real}$ were used for the computation in turn, or if the points were continuously projected onto the boundary during ongoing execution of the navigation algorithm ($\mathbf{t}_i^{virt} = \mathbf{t}_i^{real}$ and $\mathbf{g}_i^{virt} = \mathbf{g}_i^{real}$), Lloyd algorithm could be massively perturbed depending on the shape of the obstacle and result in unfavorable behavior (e.g. long paths) and undetermined configurations (e.g. oscillations). Therefore we design the algorithm in a way such that the real and virtual points remain loosly coupled until the end (we mention though that projecting continuously may have potential; an algorithm that projects the generators in each iteration step of the Lloyd algorithm is described in [12]). Whenever the position of robot i in the final configuration of a local minimum lies in the free space away from an obstacle, the robot finally succeeds in reaching its set target and $\mathbf{g}_i^{real} = \mathbf{t}_i^{real} = \mathbf{t}_i^{virt} = \mathbf{g}_i^{virt}$ holds. In the case of a convex environment, the virtual and real points simply reduce to single real points and the algorithm results in the exactly same behavior as for Voronoi coverage of convex regions, i.e. the constraints are not active.

Bug algorithms naturally provide solutions to some of the challenges of Algorithm 2. Consider the case when the final target is contained in an obstacle. TangentBug comes with a *reachability test*, where reachability is determined during the "boundary-following" behavior by just circling around the obstacle. If the exploration of the obstacle boundary is completed after one full circle without having found a leave point, the target will be unreachable. In this case, according to step 3 in Algorithm 2, the target point must be projected onto the obstacle boundary in an optimal way (see Section IV). To implement this *target projection* procedure, TangentBug can be extended with the robots checking boundary positions for optimality during "boundary-following". The optimal position along the boundary is stored in memory. Finally, projecting the point to the obstacle boundary in an optimal way simply means driving to the recorded position directly.

Once the robots have converged to a final configuration, a last Voronoi partition is computed before termination of the non-convex coverage algorithm. This last step is required because several robots might not be at the centroid of their Voronoi region due to the projection procedure (at active constraints). The final computation of the Voronoi tessellation improves the overall partition and guarantees that each dominance region assigned to a robot is at least a Voronoi region. However, notice that the final robot configuration is a constrained centroidal Voronoi configuration in general.

B. Further properties and extensions to the approach

The control strategy is general and flexible, and allows for adaptions to many different applications:

1) Offline vs. online: The approach is composed of two separate levels that are connected to each other. Level 1 and level 2 can be completely decoupled. First, a version of Lloyd algorithm runs until convergence. Second, the resulting optimal configuration is provided to the robots as input for level 2. Each robot runs its local path planner to drive to the final target position. Such a setup could be applied in a centralized off-line approach. In contrast, if level 1 and level 2 are interlaced, the interaction between the two levels enables online navigation. This is what we are mainly interested in.

2) Discrete-time vs. continuous: When level 1 and level 2 are coupled, it is a question of the frequency, sequences of level 1 and 2 follow upon each other. Classic Lloyd algorithm means that for one iteration on level 1, several loops on level 2 are executed. The Voronoi regions are not recomputed until the robots reach their current targets, i.e. the centroids of their current Voronoi regions. Under the discrete-time version of the Lloyd algorithm, the ratio between the number of iterations of loops 1 and loops 2 can be adjusted. For a ratio of unity and infinitesimal size of the iteration steps, a continuous (or quasi-continuous) version of Lloyd algorithm results. The update rates influence the overall behavior of the robots and define system requirements, such as required communication or sensing performance.

3) Known vs. unknown: Similar to Voronoi coverage in convex environments, the new control strategy can be applied to a priori known as well as unknown environments. In known environments the density function $\phi(\cdot)$ can be modeled appropriately (i.e. obstacle regions are not included in the domain Ω). The density function, the obstacle position and shape may also be unknown and must first be sensed by the robots. Extensions are toward exploration, where obstacles are learned and mapped. The density function can further be set externally to direct the robots to certain locations of special interest (formation control).

4) Selection of path planner: The modularity of the control strategy allows for the usage of any local path planner. Examples for popular obstacle avoidance methods are artificial potential fields or vector field histograms, which can be used for sensor-based navigation. In this context, further investigations can look at the method's performance, e.g. change in path lengths or time to reach a final configuration, dependent on a varying sensor range. If a map is available, a superior path planner like the A* algorithm is a good choice. For the selection of a path planner, methods with provable convergence guarantees are most preferable.

IV. Algorithm analysis

For the algorithm analysis we make following assumptions. The robots are point robots in a planar configuration space $\mathbb{R}^2$. The environment is represented by the set of reachable points Ω. Ω is bounded and the obstacles are polygonal. Both the perimeter of the obstacles and the number of obstacles are finite. The density function $\phi(\cdot)$ is defined over Ω and the distance measure $D(\cdot)$ is the Euclidean distance.

A. Convergence

We prove convergence of the control stategy for the implementation based on the Lloyd algorithm and the TangentBug algorithm for the case of planar configuration space.

Similar proofs can be given for the control strategy for other dimensions $N \neq 2$ and a case where variations on the Lloyd algorithm or a local path planner other than TangentBug are used.

***Theorem** 1 (Convergence of the control strategy):* For a non-convex environment $\Omega \subset \mathbb{R}^2$ Algorithm 1, based on the Lloyd algorithm and the TangentBug algorithm, causes the robots to converge to a constrained centroidal Voronoi configuration.

Proof: We give a proof by contradiction. Suppose that the robots do not converge to a constrained centroidal Voronoi configuration. That must be a result of: i) the TangentBug algorithm does not converge, or ii) the Lloyd algorithm does not converge. i) By Lemma 2, if TangentBug does not converge, some target point is not reachable. But that contradicts the projection properties of the control strategy (namely, that a projection of points to the environment always exists). ii) By Lemma 1, if the Lloyd algorithm does not converge, an iteration takes infinite time. But that implies i), which, as we have already shown, leads to a contradiction (namely, that TangentBug always converges to a fixed and reachable target point). ■

B. Optimality

When a target position lies outside the environment, $\mathbf{t}_i^{virt} \notin \Omega$, as soon as the corresponding virtual generator also leaves the environment, $\mathbf{g}_i^{virt} \notin \Omega$, the real position of the robot must be constrained to the environment boundary $\partial\Omega$. Let us now see how the points can be projected to the boundary in the following.

Given the control strategy in Algorithm 1 and that n robots converge to the fixed points $\mathbf{p}_i^* = \text{argmin}_{\mathbf{p}_i \in \Omega} \|\mathbf{p}_i - \mathbf{t}_i^{virt}\|$, $i \in \{1,...,n\}$, in the environment Ω, which are projections of the optimal target points $\mathbf{t}_i^{virt} \notin \Omega$, with $\mathbf{t}_i^{virt} = \mathbf{C}_{V_i}$, to the environment in the Euclidean sense. Define the final configuration vector $\mathbf{P}^* = [\mathbf{p}_1^*, ..., \mathbf{p}_i^*, ..., \mathbf{p}_n^*] \in \mathbb{R}^{Nn}$. We show in the theorem below that $\mathbf{P}^*$ minimizes the high dimensional optimization $\min_{\mathbf{P} \in \Omega^n} \|\mathbf{P} - \mathbf{T}\|$, where $\mathbf{T} = \left[\mathbf{t}_1^{virt}, ..., \mathbf{t}_n^{virt}\right]$. Furthermore we show that this implies $\mathbf{P}^*$ locally minimizes a constrained optimization problem closely related to equation (6).

***Theorem** 2 (Optimality of the control strategy):* The final configuration of the robots has the following properties:
1) The point $\mathbf{P}^*$ is closest to $\mathbf{T}$ in $\mathbb{R}^{Nn}$ in the Euclidean sense, given the projection of $\mathbf{t}_i^{virt}$ to its closest constrained point $\mathbf{p}_i^*$ in $\mathbb{R}^N$, $\forall i \in \{1,...,n\}$.
2) The final step of the control strategy (target projection) solves the constrained optimization problem of minimizing the cost function $\mathscr{H}(\mathbf{P})$ for the resulting final Voronoi partition $V = \{V_i\}_{i=1}^n$ with $\mathbf{t}_i^{virt}$ as generators.

Proof: First we prove 1) by contradiction. From the projection of the targets results that $\|\mathbf{p}_i - \mathbf{t}_i^{virt}\|$ is minimized at $\mathbf{p}_i^*$, $\forall i \in \{1,...,n\}$. Suppose that $\|\mathbf{P}^* - \mathbf{T}\|$ is not minimized. Then $\exists \|\hat{\mathbf{P}} - \mathbf{T}\|$ such that $\|\hat{\mathbf{P}} - \mathbf{T}\| < \|\mathbf{P}^* - \mathbf{T}\|$, i.e. $(\|\hat{\mathbf{p}}_1 - \mathbf{t}_1^{virt}\|^2 + ... + \|\hat{\mathbf{p}}_n - \mathbf{t}_n^{virt}\|^2)^{1/2} < (\|\mathbf{p}_1^* - \mathbf{t}_1^{virt}\|^2 + ... + \|\mathbf{p}_n^* - \mathbf{t}_n^{virt}\|^2)^{1/2}$. Substituting all $\|\hat{\mathbf{p}}_i - \mathbf{t}_i^{virt}\|$ but one by $\|\mathbf{p}_i^* - \mathbf{t}_i^{virt}\|$ leads to $(\|\mathbf{p}_1^* - \mathbf{t}_1^{virt}\|^2 + ... + \|\hat{\mathbf{p}}_i - \mathbf{t}_i^{virt}\|^2 + ... + \|\mathbf{p}_n^* - \mathbf{t}_n^{virt}\|^2)^{1/2} < (\|\mathbf{p}_1^* - \mathbf{t}_1^{virt}\|^2 + ... + \|\mathbf{p}_i^* - \mathbf{t}_i^{virt}\|^2 + ... + \|\mathbf{p}_n^* - \mathbf{t}_n^{virt}\|^2)^{1/2}$. From that it follows that $\|\hat{\mathbf{p}}_i - \mathbf{t}_i^{virt}\|^2 < \|\mathbf{p}_i^* - \mathbf{t}_i^{virt}\|^2$, $\forall i \in \{1,...,n\}$, which is a contradiction.
Now we prove 2). We can rewrite the cost function using the parallel axis theorem as $\mathscr{H}(\mathbf{P}) = \frac{1}{2}\sum_{i=1}^n J_{V_i,\mathbf{C}_{V_i}} + \frac{1}{2}\sum_{i=1}^n M_{V_i}\|\mathbf{p}_i - \mathbf{C}_{V_i}\|^2$. The first term on the right side of the equation is constant for a fixed area V_i and the mass M_{V_i} is also constant. Since $\mathbf{C}_{V_i} = \mathbf{t}_i^{virt}$, $\min_{\mathbf{P}} \mathscr{H}(\mathbf{P}) = \min_{\mathbf{P}} \sum_{i=1}^n M_{V_i}\|\mathbf{p}_i - \mathbf{C}_{V_i}\|^2 = \min_{\mathbf{P}} \sum_{i=1}^n \|\mathbf{p}_i - \mathbf{C}_{V_i}\|^2$, which is implied by the projection.

■

***Remark** 1:* The final configuration resulting from the control strategy is a constrained centroidal Voronoi configuration. We find that the costs can be further reduced, when the robots compute a last Voronoi partition V_{end} after convergence to the constrained target points $\mathbf{p}_i^*$. By recalling that Voronoi partitions act as minimizers of $\mathscr{H}$ for strictly increasing $f(\cdot)$ and CVTs as minimizers of any Voronoi tessellation V, we can derive lower and upper bounds on the costs: $\mathscr{H}(\mathbf{t}^{virt}, V) \leq \mathscr{H}(\mathbf{p}_i^*, V_{end}) \leq \mathscr{H}(\mathbf{p}_i^*, V)$.

***Remark** 2:* Theorem 2 shows the interesting result that the projection solves a constrained optimization problem. In fact, the optimal configurations to which our algorithm converges are those "orthogonality points" described in [10] (Section IV) as being desirable configurations. The algorithm in that work is proven to drive the robots to orthogonality points for a certain class of environments (convex regions with isolated holes) with considerable computational complexity. Our controller is straightforward by comparison, and is guaranteed to drive the robots to the *closest possible* orthogonality points to the true centroids. Specifically, our algorithm will not get caught in the configurations mentioned in [10] Remark 4, in which the centroid is attainable, but the robots are stuck at orthogonality points arbitrarily far from the centroid.

V. Evaluation in simulation

Next we present simulations to verify the proposed control strategy and highlight some interesting aspects. The non-convex coverage algorithm and its implementation are fully decentralized. The simulations are carried out in Matlab. The robots are point robots, the sensor range is infinite, i.e. the sensor covers the whole visible area, and the environment is assumed to be known a priori. For simplicity a uniform density function $\phi(\cdot)$ is used.

Figure 3 shows the deployment of five robots in a U-shaped environment. The robots cover the environment and converge to a final configuration, which is a centroidal Voronoi configuration in this case. The TangentBug algorithm provides the illustrated obstacle avoidance behavior and guides the robots around the obstacle's corners. The plot in Figure 3 presents the total cost $\mathscr{H}$ of the real and virtual generators for the robot configuration. It is interesting to see how the cost for the real generators approaches the cost for the virtual generators over time. The cost of the virtual generators falls off as soon as a robot reaches its current target point.

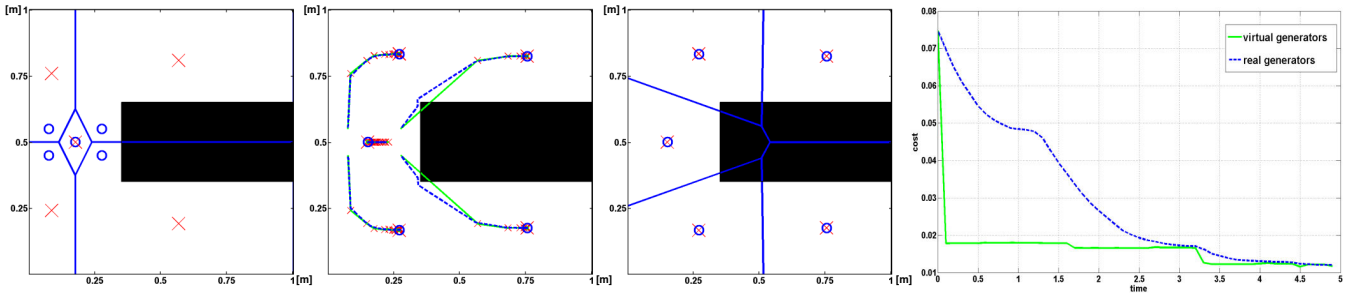

Fig. 3. Voronoi coverage in a U-shaped environment (obstacle in black). From left to right: five robots (blue circles) start from an initial position and move along the shown trajectories from one current target to the next target (red crosses) until they reach a final configuration. The next target is computed by the Lloyd algorithm as soon as a robot reaches its current target. The green lines illustrate the trajectories of the virtual generators. Two green lines intersect the obstacle's corners and show that an obstacle avoidance behavior is needed to cover this environment. The cost $\mathscr{H}$ for the configuration of the virtual and the real generators over time is shown in the plot on the right.

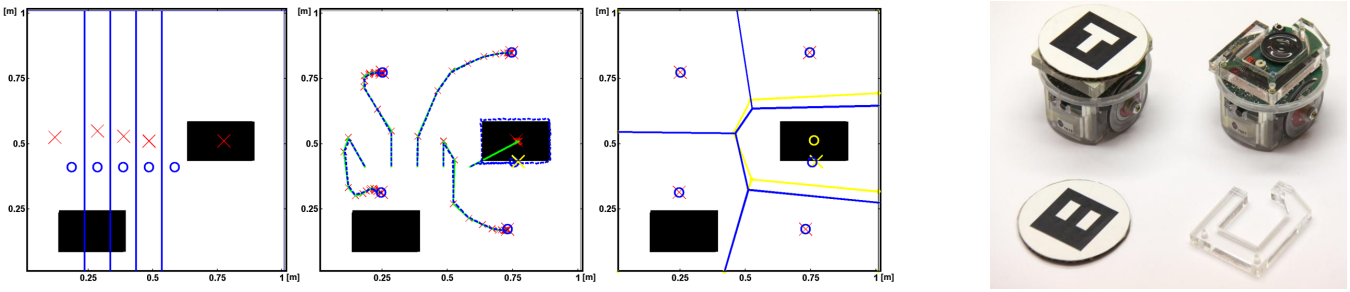

Fig. 4. Left: Voronoi coverage in an environment with free-standing obstacles (obstacles in black). The virtual generator and target of one robot lie inside an obstacle. The robot tries to reach it and starts circling around the obstacle. Finally, the target is projected to the closest point on the obstacle's boundary (yellow cross). The centroidal Voronoi tessellation (yellow) results from the virtual generators at the optimal virtual target, while the Voronoi diagram in blue is the last partition computed by the robots in a final step. Right: E-puck robot platform equipped with markers for tracking.

On the left of Figure 4, five robots cover an area with two free-standing obstacles. The virtual generator and the virtual target of one of the robots are contained in an obstacle. After the robot has explored the obstacle and completed one full cycle, it projects the virtual target to the closest point on the boundary. At the end, a last Voronoi tessellation is computed to improve the final partition (the resulting cost is reduced this way once more).

VI. Experiments with real robots

We demonstrate in the following experiment the applicability of our control strategy to real robots and non-convex environments. For the experiment we use the *e-puck* robot platform [16] (see Figure 4), a small two-wheel differential drive robot, with a diameter of 7 cm. The e-puck is equipped with a dsPIC microcontroller, different sensors (e.g. IR proximity sensors) and actuators (e.g. LEDs). The e-puck is powered by a Li-ion battery and offers a RS-232 and a bluetooth interface for communication.

We built a test setup with an overhead camera (USB-camera with resolution of 1280 x 960 pixels) to track the e-puck robots on a 1 m x 1 m ground plane in a distance of 1.35 m from the camera. Each e-puck is fitted with a marker showing a different symbol as depicted in Figure 4. We adapted the *ARToolkit* augmented reality software [17] to detect the markers and track the robots' positions and orientations. It further offers the possibility to overlay the ground plane with virtual obstacles and environment boundaries.

The images are read into a host computer and processed. The positions and orientations of the robots from the tracker are passed on to the navigation algorithm which plans the path for each of the robots for the next time step. The host computer continuously sends information over bluetooth and operates the robots by remote control. Each robot receives the commands and actuates its stepper motors, which closes the control loop. The algorithms for tracking and for the actual path planning and robot control all run in Matlab.

Figure 5 shows the initial positions and final configuration after completion of a test run with 5 robots for the U-shaped environment. The robots succeed to cover the non-convex environment. We ran seven experimental trials with the robots for the given initial positions in the U-shaped environment and emulated sensors with infinite sensor range. The paper is accompanied by a video [1] that shows one particular run of the experimental trials as well as simulations for the environments in Figure 3 and 4 and other more complex environments.

The experimental results match with the simulations. While hardware noise and tracking errors only cause small deviations, a main difference in the trajectories comes from adjustments in the algorithm to account for the non-zero size of the real robots by a safety margin along the boundary. The

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

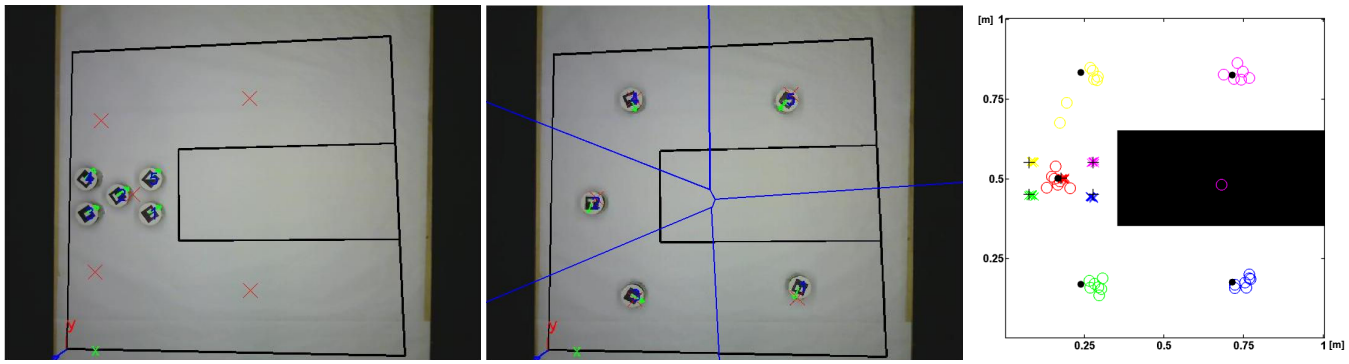

Fig. 5. Voronoi coverage in a U-shaped environment. From left to right: five robots move according to the control strategy from an initial configuration to a final optimal configuration. They avoid the obstacle and cover the non-convex environment. The plot on the right shows the initial and final configurations over the test runs. Ideal simulated positions in black (cross: initial position, dot: final position) and real experimental positions in color (cross: initial positions, circle: final positions). The magenta circle inside the obstacle shows a failed experimental run where the tracker lost the marker of one robot.

average position error over the robots and the experimental runs is 5.42 cm. The duration of one experimental run is 4.56 min in average. Though the convergence of the robots toward the final configuration was limited by the update rate of the tracking system rather than the robot platform or the control strategy itself.

VII. Conclusion

This paper presented a new control strategy to provide Voronoi coverage in non-convex environments. We analysed the problems of Voronoi coverage with non-convex environments in detail and designed a navigation algorithm based on the Lloyd algorithm and the TangentBug algorithm. Lloyd algorithm and TangentBug are interlaced and successively iterated to drive a group of robots to a final constrained Voronoi configuration. Lloyd algorithm updates the current goal position, while TangentBug runs the path planning. TangentBug is modified with a projection procedure to constrain outlying target points to the environment. The constrained points resulting from the projection were shown to be solutions to the constrained optimization problem of finding a robot configuration for the final Voronoi partition of minimal cost. We proved both convergence and optimality of the proposed control strategy, by applying the concept of virtual generator and target points to multi-robot path planning. The approach has been evaluated in simulations and physical experiments with a team of networked robots. We further demonstrated that the approach is general and offers many interesting extensions, such as applying Voronoi coverage to unknown environments, using a different path planner better suited to a specific application or modifying the coupling between Lloyd algorithm, path planner and projection procedure.

In our future work we are interested in Voronoi coverage in combination with sensor-based navigation in unknown and non-convex environments. A direction will be to extend the control strategy for adaptation to and learning or mapping of the obstacles. Another step is to implement Voronoi coverage directly on the robot platform.

References

[1] D.W. Gage, "Command Control for Many-Robot Systems", in *the Nineteenth Annual AUVS Technical Symposium AUVS-92*, reprinted in *Unmanned Systems Magazine*, vol. 10, no. 4, pp. 28–34, 1992.

[2] 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.

[3] M. Schwager, D. Rus and J. Slotine, "Decentralized, Adaptive Coverage Control for Networked Robots", *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[4] 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.

[5] A. Okabe and A. Suzuki, "Locational optimization problems solved through Voronoi diagrams", *European Journal of Operational Research*, vol. 98, no. 3, pp. 445–456, 1997.

[6] A. Ganguli, J. Cortés and F. Bullo, "Multirobot rendezvous with visibility sensors in nonconvex environments", *IEEE Trans. on Robotics*, vol. 25, no. 2, pp. 340–352, 2009.

[7] A. Ganguli, J. Cortés and F. Bullo, "Distributed Coverage of Nonconvex Environments", in *Proc. of the NSF Workshop on Future Directions in Systems Research for Networked Sensing*, Lecture Notes in Control and Information Sciences, pp. 289-305, 2007.

[8] N. Ayanian and V. Kumar, "Decentralized Feedback Controllers for Multi-Agent Teams in Environments with Obstacles", in *2008 IEEE Int. Conference on Robotics and Automation*, pp. 19–23, 2008.

[9] C. Caicedo and M. Žefran, "Performing coverage on nonconvex domains", in *Proc. of the 2008 IEEE Multi-Conference on Systems and Control*, pp. 1019–1024, 2008.

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

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

[12] 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.

[13] I. Kamon, E. Rimon and E. Rivlin, "A new range-sensor based globally convergent navigation algorithm for mobile robots", CIS - Center of Intelligent Systems, Computer Science Dept., Technion, Israel, 1995.

[14] I. Kamon, E. Rimon and E. Rivlin, "Tangentbug: A Range-Sensor-Based Navigation Algorithm", *International Journal of Robotics Research*, vol. 17, no. 9, pp. 934–953, 1998.

[15] M. Pavone, E. Frazzoli and F. Bullo, "Distributed policies for equitable partitioning: Theory and applications", in *Proc. of the 47th IEEE Conference on Decision and Control*, pp. 4191–4197, 2008.

[16] 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.

[17] 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.