2012 8th IEEE International Conference on Distributed Computing in Sensor Systems

On the Feasibility of Mass-Spring-Relaxation for Simple Self-Deployment

Juergen Eckert*, Hermann Lichte†, Falko Dressler‡ and Hannes Frey§
*Dept. of Computer Science, University of Erlangen, Germany
†net mobile AG, Duesseldorf, Germany
‡Institute of Computer Science, University of Innsbruck, Austria
§Dept. of Computer Science, University of Paderborn, Germany
juergen.eckert@cs.fau.de, hermann.lichte@net-m.de, falko.dressler@uibk.ac.at, hannes.frey@upb.de

***Abstract*—Self-deployment describes the task of spreading an autonomously moving swarm of mobile robots over a given area. All these robots have to move to locations such that the set of robot locations satisfies a desired property. In this work, we describe a fully distributed deployment algorithm executed locally at each robot. The approach requires only few local information per node: the distances and very coarse angular information to immediate neighbors. It has been developed for use on small robots with very restricted memory, communication, and processing capabilities. In this paper, we specify the algorithm and evaluate it in an empirical study. This includes both simulation studies and real testbed experiments. For the testbed, we consider two different platforms: ground moving robots and aerial robots. The results of our simulations show that our local deployment rules achieve almost globally optimal results. The testbed study supports and substantiates our simulation study and shows as a proof of concept that our algorithm works both with real ground based and aerial based robot swarms.**

I. Introduction

Autonomously acting robots can be a very helpful tool whenever direct human action is economically impractical, unsafe, or even impossible [1]. For example, a flying drone can perform autonomous measurements very economically. Moreover, autonomous robots can prevent life-threatening situations in exploration, firefighting and first response operations, or cleanup efforts in hazardous areas. Furthermore, such robots can be used for exploration of unknown territories, an abyssal plain, or, eventually, a foreign planet.

A complex observation or manipulation task may require multiple robots to jointly collaborate on its execution. For instance, a single robot might be too small to surmount an obstacle or to pull an object. Collaboratively, swarm robots may be able to achieve the task [2]. Another example is monitoring of large areas. The sensors of a single robot may be able to only cover a small fraction. In a team, robots can evenly deploy itself over the area and monitor the whole area at once.

In this work, we consider the problem of covering an area with a set of robots. This is also referred to as the *self-deployment problem* [3]. Starting from a set of arbitrarily placed robots, the robots have to relocate such that the whole swarm eventually covers a given area according to some metric. In this paper, we expect the robots to spread over the area in a uniform way, i.e., each robot has approximately the same number of neighboring robots.

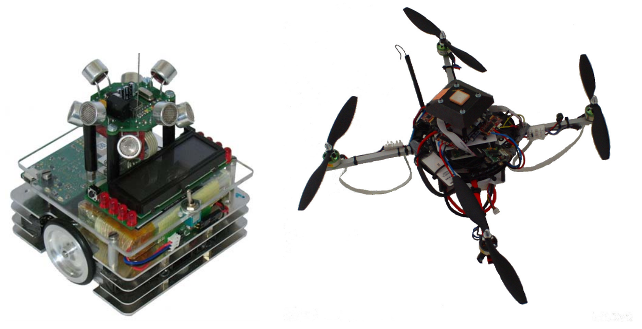

(a) Land robot (b) Aerial robot

Fig. 1. Hardware platforms

The target for our approach is a large scale robot swarm where cost per unit is a crucial factor. Each individual robot can only be equipped with very limited computational, communication, storage and sensing capabilities. Thus, all utilized algorithms, including self-deployment, have to be very resource efficient.

In our approach, which is based on our previous work in [4], robots get informed only about the nodes in their immediate neighborhood without any additional relays. Information required for each node is just the distance and a coarse angular estimate towards its neighbors. Based on this information, each robot follows a small, computationally simple, set of movement updates. These locally computed updates emerge to a globally uniformly deployment of robots.

In this paper, we evaluate our approach by means of simulation as well as using prototype implementations on two different hardware platforms depicted in Figure 1. Besides of totally different movement characteristics, they also have disjoint neighbor sensing techniques. The ground robots sense their neighbors passively by Ultrasound (US) beacons, whereas the aerial drones are equipped with GPS and exchange their position information via radio.

II. Related Work

Typical self-deployment mechanisms require some means to assess a node's current location with respect to its neighboring nodes. Such information can either be determined in a relative or an absolute manner. In the relative case, distances or relative bearings towards neighboring nodes are determined. Distance information can for instance be determined using infrared/laser scanners. In approaches based on absolute information, each node knows its own geographic location as well as that of its neighbors. Such information can be made available by GPS or some local indoor positioning infrastructure.

Deployment based on geographic location information can be classified in Voronoi based and tessellation based approaches. Voronoi based approaches have for example been described in [5]–[7]. The general idea is to compute a Voronoi diagram over the set of node positions. The resulting Voronoi regions are then used to decide if a node can stay put or if it should move and where it should move. For instance, in [7], a node whose surrounding Voronoi region is not completely covered by its sensing range, always moves to the most distant vertex of that region. If the region is completely covered the node stays put.

Tessellation based approaches partition the deployment region with a grid of regular polygons. Approaches based on tessellations have for example been described in [3], [8], [9]. In those approaches, tessellation is used to discretize the deployment problem. Nodes will align themselves towards the tessellation vertices and start to "jump" along the tessellation vertices according to some local rules. In [9], for instance, nine local greedy advance and rotation rules have been described, which assure that after a finite number of movement rounds the tessellation vertices, which optimally cover a given point of interest, are occupied by exactly one mobile node.

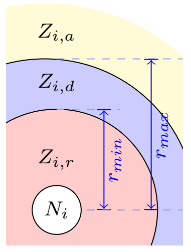
Finally, in vector based deployments, a node computes for each neighbor a vector which is either pointing to or pointing away from this neighbor. The sum of all these vectors determines the node's current movement vector. Approaches based on such vector addition have for example been described in [10]–[16]. The approaches differ in the way how attractive or repulsive vectors are determined from node distances and how a stable local equilibrium is reached by friction forces.

III. Spring Based Deployment Approach

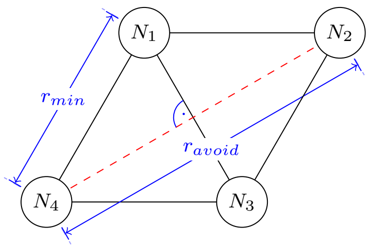
For the following formal considerations a network is represented as a connected graph $G = (V, E)$ with a vertex set V and an edge set E. A node $N_i \in V$ represents the i^{th} robot. An edge (N_i, N_j) exists in E if robot i can sense robot j Robot j is called a neighbor of i in this case and *vice versa*.

We consider a mapping $p : V \rightarrow \Re^n$ which maps each node into a Euclidean space. The value $p(N_i)$ represents the physical positions of the i^{th}-robot. In the following we consider deployment in two dimensional space ($n = 2$). However, only minor parameter changes are required to solve the three dimensional deployment problem.

As shown in [12], [17], [18], an equilateral triangle tessellation can keep connectivity and maximize the coverage area without coverage gap. Here we want to show how to efficiently achieve this structure using only a small set of rules, computations and information. Additionally, we assume that no global knowledge is available, i.e., computations and actions must be performed locally.



(a) Zones around a node N_i



(b) Base structure

Fig. 2. Basic geometry

Recently, we introduced a nature inspired approach [4], which is based on the principles of Mass-Spring-Relaxation (MSR) [19]. The physical model is as follows. A node N_i is represented by a mass point m_i and an edge (N_i, N_j) by a linear elastic spring with a desired/relaxed length of d_0. If two mass points m_i, m_j are too close (i.e., $\|p(N_i) - p(N_j)\|_2 < d_0$, where $\|\cdot\|_2$ is the Euclidean distance) then the affecting force of the spring is pulling them apart. Otherwise, if $\|p(N_i) - p(N_j)\|_2 > d_0$, the spring is pulling them together. The force affecting a mass point m_i can be computed according to Hooke's law (see Equation 1), where $k_{i,j}$ is the spring constant (usually normalized to 1), $\vec{e}_{i,j}$ characterizes the unit vector from node N_i to node N_j, and $d_{i,j} = \|p(N_i) - p(N_j)\|_2$ represents the corresponding measured distance.

$$\vec{F_i} = \sum_{N_j \in Z_{i,s}} \vec{F}_{i,j} = \sum_{N_j \in Z_{i,s}} -\vec{e}_{i,j} k_{i,j} (d_{i,j} - d_0) \qquad (1)$$

MSR-based deployment without additional provision will not necessarily result in an equilateral triangle deployment. Our approach extends MSR deployment with additional rules defining which neighbors of node N_i are used to compute the force affecting node N_i. Each node considers a small circle with radius $r_{\min}$ and a larger circle with radius $r_{\max}$ (see Figure 2(a)). Both circles define the depicted restricted $Z_{i,r}$, desired $Z_{i,d}$, and attractive zones $Z_{i,a}$ of a node N_i. Each node follows a set of relocation rules with the goal to keep the restricted area empty and the desired area filled with a certain minimum number of nodes. The attractive zone is considered only if the restricted and desired zones are empty. The rationale behind the latter is to contract nodes together in case of a sparse initial start deployment. This concept is described in detail in the following.

A. Procedure

Algorithm 1 displays the main deployment loop. At the beginning a node N_i needs to sense all of its neighbors. A wide variety of measurements is feasible. In this paper, we introduce two different hardware platforms with completely different sensing techniques. In the end the gathered information should provide a distance estimate as well as a rough estimate

Algorithm 1 Deployment algorithm for node N_i

```
  while deploying do
    do measurements → $Z_i := \{N_x | \forall (N_i, N_x) \in E\}$
    do grouping → $Z_{i,r} \subseteq Z_i; Z_{i,d} \subseteq Z_i; Z_{i,a} \subseteq Z_i$
    do selection → $Z_{i,s} := Z_{i,x};\ x \in \{r, d, a\}$
    move → $p(N_i)_{t+\Delta t} = f(p(N_i)_t, \vec{F}_i(Z_{i,s}), \Delta t)$
  end while
```

of the angle of arrival ($\pm 45°$ is sufficient) between a node N_i and any other neighboring node N_x.

Subsequently these neighbors are grouped into three different sets according to their distances (Equations 2-4). The radii r_{min}, r_{max} and r_c will be explained in detail in Section III-B.

$$Z_{i,r} := \{N_x \mid 0 < \|p(N_i) - p(N_x)\|_2 < r_{min}\} \quad (2)$$

$$Z_{i,d} := \{N_x \mid r_{min} \leq \|p(N_i) - p(N_x)\|_2 \leq r_{max}\} \quad (3)$$

$$Z_{i,a} := \{N_x \mid r_{max} < \|p(N_i) - p(N_x)\|_2 < r_c\} \quad (4)$$

A node's next movement decision is based on these sets and the decision table shown in Table I. A node checks the table row by row to find the first valid guard. The force vector $\vec{F_i}$ (Equation 1) is then computed using the neighbor set $Z_{i,s}$ specified in that row as an input. The actual movement (Equation 5) yields to this virtual force $\vec{F_i}$. The passed time per step Δt as well as $\frac{1}{2m_i}$ are constant factors. In case of no neighbors $Z_{i,s} = \emptyset$ the node tries to find a network by randomly choosing $p(N_i)_{t+\Delta t}$.

$$p(N_i)_{t+\Delta t} = \frac{1}{2m_i} \overrightarrow{F}_i \Delta t^2 + p(N_i)_t \quad (5)$$

The second rule of Table I allows two different ways how the deployment algorithm reacts in case a sufficient amount of neighbors is in the desired set ($\lambda \leq |Z_{i,d}|$). If the goal is to deploy a system in a shortest possible period (in terms of active movement), to use as little energy as possible, and to have a stop/suspend criteria, then the robots should stop their movement as soon as rule 2 can be applied (case B). As long as the neighbors of node N_i do not enter $Z_{i,r}$ or leave $Z_{i,d}$ (preferably also stopped), node N_i will not move again and thus conserves energy. This is the case for the ground robot scenario (Section V-A). For the flying robot scenario (Section V-B) the energy costs for the movement are the same as for the hovering. Therefore, the movement can be continued in rule 2 and a more regular grid is formed (case A).

TABLE I
PLACEMENT DECISION CONSTRAINTS

Guard	Action	Use nodes ($Z_{i,s}$)
$\|Z_{i,r}\| > 0$	move	$Z_{i,r}$
$\lambda \leq \|Z_{i,d}\|$	(A) move or (B) place found	(A) $Z_{i,d}$ or (B) $\emptyset$
$\|Z_{i,a}\| > 0$	move	$Z_{i,a}$
else	random move	$\emptyset$

B. Parametrization

The factor $\frac{k_i \Delta t^2}{2m_i}$ for the movement (Equations 5 and 1) is constant and system dependent. It defines how fast the system reaches a steady state. As a rule of thumb: a good starting value is 0.5 (each variable set to 1). If nodes overshoot (due to measurement and/or positioning inaccuracies) and continuous oscillations remain in the system, this value should be lowered.

Four additional parameters need to be defined such that a regular grid structure is achieved: r_{min}, r_{max}, λ and d_0. The first two parameters divide the space around a node N_i into three sections (see Figure 2(a)). The key idea is to cover as much ground as possible. By definition, the algorithm does not allow neighbors to be within the restricted set $Z_{i,r}$. Therefore, r_{min} which defines $Z_{i,r}$ also indirectly defines the system costs in terms of nodes required per unit area.

An optimal solution would be to set r_{min} very close to the communication/measurement range r_c. However, there are some other constrains like convergence time that also need to be fulfilled. The convergence is affected by the belt around node N_i defined by r_{min} and r_{max}. For example, for an optimal regular triangle tessellation those radii should be very close since in such structure all node distances are equal. However, in the real world this is not applicable. The algorithm may never find a steady state, may take too much time, or obstacles might inhibit convergence.

The upper bound of r_{max} is defined as follows. Figure 2(b) depicts two adjacent regular triangles (base structure). Consider the worst (cost inefficient) case: all node distances are r_{min}. It must be avoided that node N_2 connects to node N_4. If those nodes connect, they start to form a square grid. This approach is without extensions rather limited applicable for variable relaxed node distances. Also a square grid structure has the worse area-cost ratio. To avoid this the following condition must hold ($r_{avoid} = \sqrt{3}\ r_{min}$): $r_{max} < \sqrt{3}\ r_{min}$.

To achieve the maximal coverage, r_{max} can be set to r_c. If so rule 3 of Table I can also be ignored. However, mostly the sensing gets non-linear at the boards of the detection range or packages get dropped more frequently. To counteract this r_{max} should be set to the maximum linear detection range. The outer domain still gets evaluated if needed.

The desired spring length d_0 of Equation 1 is defined as:

$$d_0 = \frac{r_{max} + r_{min}}{2} \quad (6)$$

That way nodes always try to reach the center of the desired belt to maximize the tolerance range.

In a regular hexagonal grid cell structure the neighbor connectivity is in the range of 3 to 6. The inner nodes all have 6 neighbors. Nodes on a corner 3. Therefore, λ which represents the minimum connectivity must be set to 3. For small grid sizes $|V| < 7$, as depicted in Figure 2(b), λ needs to be reduced to 2 to find a steady state. However a long chain of nodes instead of a grid structure might result from this. To counteract this additional techniques are required as introduced by Casteigts *et al.* [20]. Another alternative is to start all nodes at the same place. That way a circular network is created.

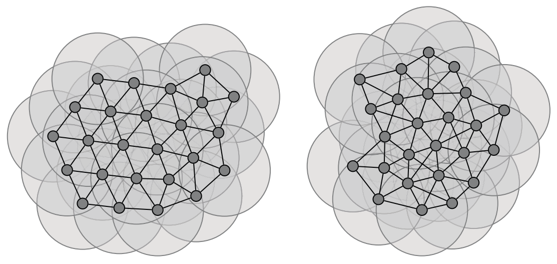

(a) Continuous movement (b) Stop citeria

Fig. 3. Exemplary simulation outcome

IV. Simulation Study

Before conducting real experiments, we analyzed the approach in the JBotSim simulator [21]. We are mainly interested in the behavior of the algorithm, thus we neglect measurement and communication errors. Figures 3(a) and 3(b) depict exemplary outcomes for a network size of $|V| = 25$. The gray circle around each node indicates the desired node distance d_0. An edge between a node pair depicts a virtual spring. Nodes in the former plot continuously move (case A) and therefore the shape is more regular than the latter one (case B).

We evaluated the convergence time using the following parameters: $r_{min} = 3.9\,\text{m}$, $r_{min} = 6.1\,\text{m}$, $r_c = 7.0\,\text{m}$. A node can have a maximum speed of $0.5\,\text{m}\,\text{s}^{-1}$. We conducted two different time-evaluating experiments using a network size of $|V| = n$. We repeated each experiment 200 times to achieve statistically significant results. The following figures show the *self-displeased node ratio* that we defined as follows: A node N_s which cannot apply rule 2 in Table I is considered to be *self-displeased* $N_s \in V_d$. It moves until rule 2 can be applied. The ratio is then defined as a quotient of the amount of nodes in this moving state and the total number of nodes: $SD = \frac{|V_d|}{|V|}$.

In the following, we plot the median (50 % quantile) of the measurement samples. The error bars indicate the stable co-domain (25 % and 75 % quantiles).

In the first set of experiments (Figure 4(a)), the initial positions of all nodes are uniformly distributed in an unbounded area of $80\,\text{m} \times 60\,\text{m}$. The nodes need to find each other and form a network. In general, the formation requires more time if less nodes are on the field. For a sparse node density, more than one connected network may be created as no root node is specified. Uncovered areas are possible.

In the second set of experiments (see Figure 4(b)), all nodes have nearly the same position when the simulation starts. Here, smaller networks converge fast because the required movements of the individual nodes are shorter. The limiting factor is the velocity. The outer nodes very quickly start to move away from the center with maximum speed.

Both variants of the algorithm (rule 2 in Table I) have similar deployment times. Introducing the stop criteria (case B) slows down the deployment a bit, but still is negligible for real world experiments.

Finally, we evaluated the covered area against other approaches which lead to a perfect regular hexagonal deployment. For example, Li *et al.* [9] require a set of 9 complex rules to achieve such deployment. Figure 4(c) depicts the area ratio $AR = \frac{A_{vector}}{A_{opt}}$ between the covered area A_{vector} of our approach and the covered area A_{opt} of a regular hexagonal deployment (optimal solution) for different network sizes $|V|$.

For the optimal solution, we set the node distance to d_0 (see Equation 6). We performed simulations for both algorithm variants. Again, we conduct multiple runs per network size and the error bars indicate the stable co-domain. For the continuous movement (case A) we observe an area defect of less than 4 %. The reason for the significant drop in the second case (using stop criteria) is as follows: the belt width of the desired zone $Z_{i,d}$ is set to $\pm 22\,\%$ of d_0. All nodes start at nearly the same location. After spreading out and reaching approximately 80 % of d_0, the nodes can execute rule 2 in Table I and hence they stop. If the belt width is reduced the ratio would also reach the result of the continuous movement. However, an insufficient belt width in combination with measurement or positioning errors might prohibit a steady state solution. The benefit here is that the individual nodes do not continuously move. Therefore, they can shut down the motors frequently and conserve energy.

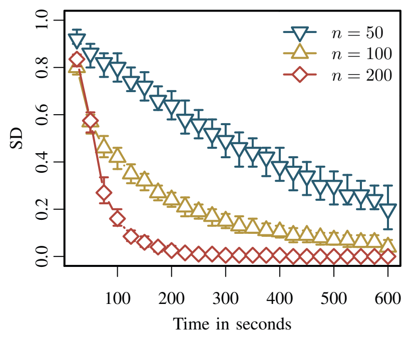
V. Hardware Implementations

We implemented our approach on two different hardware platforms, ground swarm and aerial swarm, each with their specific measurement and movement techniques.

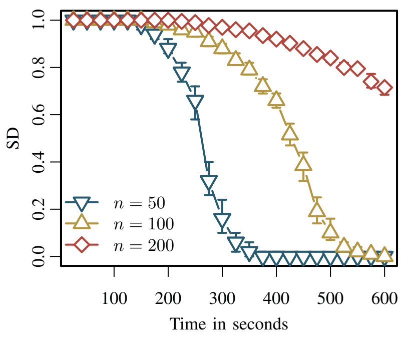
A. Ground Swarm

The ground swarm consists of mobile two-wheel robots [22] depicted in Figure 1(a). We use US based Time of Flight (ToF) measurement hardware to determine the distance to neighboring nodes. The hardware also estimates the angle of arrival ($\leq 45°$; distance dependent). No radio communication is required. The robots do not know their absolute position, thus all decisions are made relatively to the neighbors. A key goal of the battery powered system is to stay operational for as long as possible. Therefore, it is necessary to find a valid position as fast as possible and to disable motors and driver-stages afterwards. This is a classical field of application for the algorithm including a stop criteria.

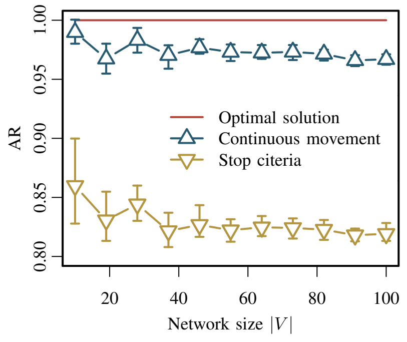
Measurement disturbances caused by measurement errors or obstacles are not always a drawback. They sometimes can be exploited to solve restricted situations. Consider the following constructed example depicted in Figure 5. Five nodes have been well distributed before. Let us assume that within this area, some node does not find a valid solution anymore due to spacing issues, e.g., $r_{min} = 0.7\,\text{m}$. In this case, the algorithm is designed such that one of the outer nodes re-activates the motors (as the inner node is approaching too close) to give room for the deployment of the inner one. However, if an application such as [4] prohibits this then in a noiseless environment, the inner robot would continue driving until the battery is depleted. The algorithm does not detect such a situation due to its simplicity. Measurement disturbances may lead to an incorrect computation of the force vector. Similarly, obstacles occasionally detected on the current path of the robot may



(a) Deployment process (initially uniform distributed)



(b) Deployment process (initially not distributed)



(c) Optimal vs. vector deployment

Fig. 4. Simulations

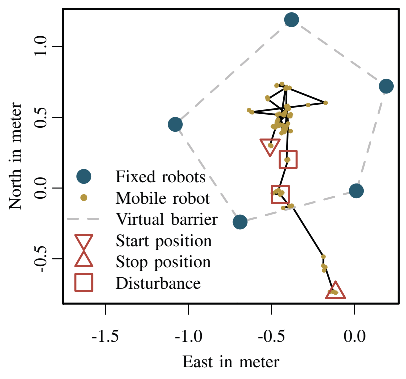


Fig. 5. Exemplary run using fixed nodes

entail an alteration of the trajectories planed by the deployment algorithm. Although not intended, these real-world effects can help to overcome such restricted situations.

As depicted in Figure 5, the trajectory starts within a fully covered area. Each fixed node is periodically emitting a US pulse. The mobile node converts these pulses into distances and angles. Subsequently, the node yields to the affecting forces and moves to the center. However, we physically disturbed the reception of these signals in approximately $5\,\%$ of the cases. Following the two positions denoted by a square in Figure 5, the algorithm made significant false decisions. The upper one brings the mobile robot close to the direct line between the two lower fixed robots. This poses a virtual force barrier that the robot cannot breach. However, close to this line (lower square) the measurement accuracy of the angle makes it impossible to decide whether the robot is above or below this line. Therefore, it might happen (as depicted) that it *accidentally* breaches the barrier and finds a suitable location outside. The experiment duration was $4\,\mathrm{min}$.

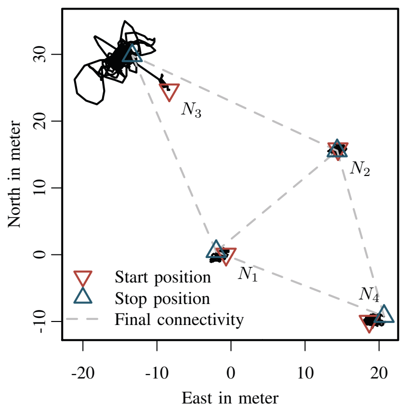
In general, if an application restricts the movement after the deployment, it is highly recommended to detect and to solve such situations in a different way. E.g., by performing a major random move every n^{th} iteration or by choosing a more complex approach. Otherwise it might take too long to solve the issue by purely relying on measurement disturbances.

B. Aerial Swarm

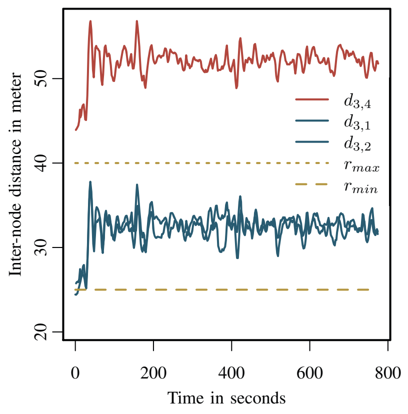
Figure 1(b) shows a single aerial robot. Its core component is a fully equipped “MikroKopter” which features a GPS position-hold functionality out of the box. It can retain the copter in a sphere with a radius of a few meters (strongly dependent of GPS accuracy and wind speed). In contrast to the ground robot experiments, the copters use GPS and radio communication to learn their absolute positions as well as the positions of their neighbors. Relative distances and angles are computed upon this base. Hovering in the air drains as much energy as slow movements. Thus, for the deployment algorithm of the flying robots a continuous movement (case A of decision table I) can be assumed.

Figure 6(a) depicts an initial exemplary experiment run. Four copters are forming a network. All platforms had the same altitude. Due to the small network size $|V| = 4$, the minimal connectivity needed to be lowered to $\lambda = 2$. The remaining parameters are defined as follows: $r_{min} = 25\,\mathrm{m}$; $r_{max} = 40\,\mathrm{m}$; $r_c = 100\,\mathrm{m}$. Light air was present during the test run. The black lines are indicating the trajectories over time. To show the accuracy of a single system the robots N_1, N_2 and N_4 were set to a fixed position. The initial setup is defined so that it matches Figure 2(b). The inter-node distances were approximately r_{min}. Within the first $30\,\mathrm{s}$ the mobile node N_3 finds a valid position (d_0 away from N_1 and N_2) and stayed there. Figure 6(b) shows the distances of interest over time. It can be seen that the distances to connected nodes are oscillating around $d_0 = \frac{r_{max}+r_{min}}{2}$. As desired, no connection to node N_4 was generated ($d_{3,4} > r_{max}$). Numerical values indicate that more than $50\,\%$ of the measured positions had an error of less than $1\,\mathrm{m}$.

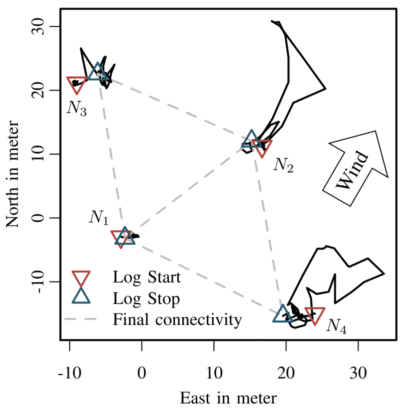
Figure 6(c) depicts a second exemplary test run. The weather was more gusty (up to approx. $25\,\mathrm{km\,h^{-1}}$ wind speed). This plot shows the trajectory of the robots during position maintenance. Robot N_1 needed to be at a fixed position. Otherwise the whole system starts to drift at wind speed. The others were operating normal. It can be seen that two equilateral triangle (base structure) are formed. The depicted duration of the experiment is $2\,\mathrm{min}$. At the middle of the depicted time interval the robots N_2 and N_3 could not hold against the wind force and drifted



(a) Exemplary position jitter (one mobile)



(b) Inter-node distances over time for exemplary run



(c) Exemplary position jitter (three mobile)

Fig. 6. Aerial swarm experiments

away (robot N_3 is located at a lee side). However the network stayed connected, adjusted and found the correct formation after the gust again.

VI. Conclusions

We presented a simple self-deployment mechanism for autonomous robot swarms based on a small set of decision rules, vector addition, and scalar multiplication. This allows for an implementation on simple embedded computing devices. Though the rules are simple, they show to be effective in terms of optimal node deployment. In this work, we emphasized on nodes deployed in regular triangle tessellation pattern and we showed by simulation that this pattern is closely resembled. In the simulation study we have seen that the areal defect in comparison to more complex solutions is negligible small. To validate our concepts in realistic scenarios, we realized our approach under real world conditions with different embedded robot devices. In conclusion, it can be said that we managed even the dynamic use case of coordinating flying quadcopters. The proposed algorithmic architecture can easily be used in similar scenarios. Most importantly, the algorithms perform extremely satisfying on embedded ultra low resource devices, which opens a variety of application scenarios.

References

[1] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for swarm robots," in *IEEE/RSJ IROS*, July 1993, pp. 441–447.

[2] R. O'Grady, R. Gro, A. Christensen, and M. Dorigo, "Self-assembly strategies in a group of autonomous mobile robots," *Autonomous Robots*, vol. 28, no. 4, pp. 439–455, 2010.

[3] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri, "Snap and spread: A self-deployment algorithm for mobile sensor networks," in *IEEE DCOSS*, Santorini Island, Greece, June 2008, pp. 451–456.

[4] J. Eckert, R. German, and F. Dressler, "ALF: An Autonomous Localization Framework for Self-Localization in Indoor Environments," in *IEEE DCOSS*, Barcelona, Spain, June 2011, pp. 1–8.

[5] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 35, no. 1, pp. 78–92, 2005.

[6] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[7] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.

[8] S. Yang, M. Li, and J. Wu, "Scan-based movement-assisted sensor deployment methods in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1108–1121, 2007.

[9] X. Li, H. Frey, N. Santoro, and I. Stojmenović, "Strictly localized sensor self-deployment for optimal focused coverage," *IEEE Transactions on Mobile Computing*, 2011.

[10] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi, "A distributed sensor relocatlon scheme for environmental control," in *IEEE MASS*, Pisa, Italy, October 2007.

[11] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *DARS*, 2002, pp. 299–308.

[12] M. Ma and Y. Yang, "Adaptive triangular deployment algorithm for unattended mobile sensor networks," *IEEE Transactions on Computers*, vol. 56, no. 7, pp. 946–847, 2007.

[13] G. Tan, S. A. Jarvis, and A.-M. Kermarrec, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," in *ICDCS*, Washington, DC, 2008, pp. 429–437.

[14] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, pp. 61–91, 2004.

[15] J. McLurkin and J. Smith, "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots," in *DARS*, 2007, pp. 399–408.

[16] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *IEEE/RJE ICRA*, New Orleans, LA, April 2004, pp. 165–171.

[17] H. Zhang and J. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, no. 1-2, 2005.

[18] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *ACM MobiHoc*, Florence, Italy, May 2006, pp. 131–142.

[19] A. Howard, M. J. Mataric, and G. Sukhatme, "Relaxation on a Mesh: a Formalism for Generalized Localization," in *IEEE/RSJ IROS*, Maui, HI, October 2001, pp. 1055–1060.

[20] A. Casteigts, J. Albert, S. Chaumette, A. Nayak, and I. Stojmenovic, "Biconnecting a Network of Mobile Robots Using Virtual Angular Forces," in *IEEE VTC*, Ottawa, Canada, September 2010, pp. 1–5.

[21] A. Casteigts, *The JBotSim Library*, e-Print (arXiv:1001.1435), 2010, http://arxiv.org/abs/1001.1435.

[22] J. Eckert, K. Koeker, P. Caliebe, F. Dressler, and R. German, "Self-localization Capable Mobile Sensor Nodes," in *IEEE TePRA*, Woburn, MA, November 2009, pp. 224–229.