# Coordination of Mobile Agents for Simultaneous Coverage

Petra Mazdin and Bernhard Rinner
{petra.mazdin,bernhard.rinner}@aau.at

Institute of Networked and Embedded Systems, University of Klagenfurt, Austria

**Abstract.** Simultaneous environment coverage represents a challenging multi-agent application, in which mobile agents (drones) must cover surfaces by simultaneously capturing images from different viewpoints. It constitutes a complex optimization problem with potentially conflicting criteria, such as mission time and coverage quality, and requires dynamic coordination of agent tasks. In this paper, we introduce a decentralized coordination method, adaptive to a dynamic and a priori unknown 3D environment. Our approach selects the role an agent should take on and coordinates the assignment of agents to their computed viewpoints. Our main goal is to cover all detected objects in the environment at a certain quality as soon as possible. We evaluate the methods in AirSim in different setups and assess how the proposed methods respond to dynamic changes in the environment.

**Keywords:** Multi-agent system · simultaneous coverage · drones · viewpoint constellations · market-based task assignment · AirSim simulator.

## 1 Introduction

Mobile robots represent a prototypical example of a multi-agent system, and we have witnessed their tremendous progress in research and applications over the last decades. Collaborative aerial robots or multi-drone systems (e.g. [21,25]) are a particularly challenging research field due to their flexibility, scalability and resource limitations.

This paper deals with *simultaneous coverage* of unknown environments which represents an important problem for multi-drone systems. In various applications, such as monitoring, inspection, 3D reconstruction, and depth measurements, drones with onboard cameras autonomously move in the environment to capture imagery of objects of interest with sufficient quality [15]. In case of dynamic environments, the capturing time of images is highly relevant, and estimation of the state of the environment or the objects of interest may become uncertain if the time lag of the individual image capturing is too large. Simultaneous coverage mitigates this problem by requiring concurrent image capturing from $k$ different viewpoints and thus simplifies multi-view image analysis.

In our approach, objects of interest must be first detected and then covered by simultaneously captured images from $k$ different viewpoints. Fig. 1 depicts

different stages of such coverage mission where drones explore the initially unknown environment (gray area) to detect objects (pink cuboids). Once an object has been detected, it needs to be observed in order to abstract its shape and to compute the required viewpoints (constellations) for the coverage. Finally, dynamic teams are formed and move along the paths to the assigned viewpoints. When the required viewpoint locations are reached, the drones simultaneously capture images with overlapping field of view (yellow area) and continue the mission. Dynamic coordination of tasks and paths is crucial, since simultaneous coverage constitutes a complex optimization problem with changing knowledge about the state of the environment.

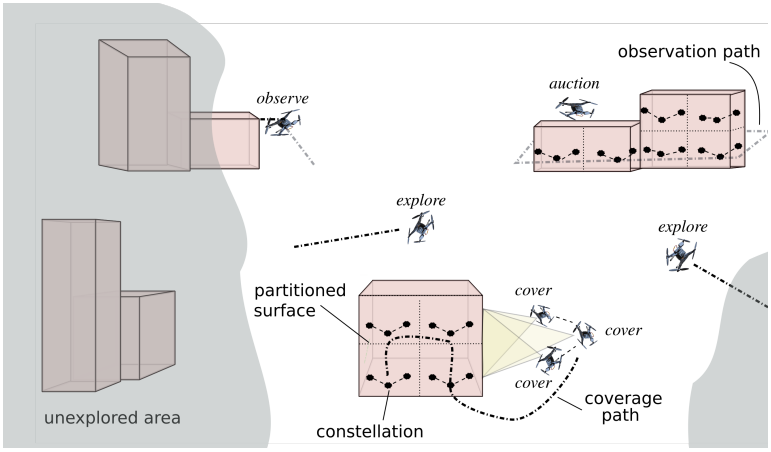



Fig. 1: Sketch of a simultaneous coverage mission where drones explore the environment, observe detected objects, compute constellations, and navigate along the constellations in dynamically assigned teams (here for $k = 3$ drones).

Weyrer and Rinner [24] introduced a path planning algorithm and a model-predictive controller for a fixed team of two drones, and Mazdin and Rinner [16] proposed a market-based drone assignment for coverage in simple, a priori known 2D environments. The contribution of this paper can be summarized as follows: First, we expand the simultaneous coverage algorithm to 3D unknown environments where drones autonomously take on different roles. Second, the computation of the required viewpoints explores the tradeoff between achieved coverage quality and coverage area. Third, we introduce an adaptive market-based coordination approach for dynamic task assignment. Finally, we perform a simulation study of our approach using the multi-drone framework AirSim [22].

The paper is organized as follows: Section 2 briefly compares our contribution with the state of the art. Section 3 provides the formal problem definition, and Section 4 describes our approach. Section 5 discusses the achieved results, and Section 6 summarizes the contribution and discusses potential future work.

## 2 Related Work

Even with small-scale drones, leveraging onboard cameras is widely used for various inspection tasks of an object's surface, and the approaches differ whether a priori knowledge of the environment or the object's shape is considered. Heng et al. [13] propose an algorithm for simultaneous exploration and coverage in an unknown environment, whereas Bircher et al. [3] address the exploration of an unknown environment and extend their work to include the inspection of surfaces in [4]. An alternative approach to a similar exploration path planning problem can be found in [26]. Galceran and Carreras [10] survey the problem of coverage path planning and its applications.

Stereo coverage is a well-known topic of arranging two cameras with overlapping fields of view (FoV) [4,9,13]. Gallup et al. [11] introduce the concept of altering the baseline and resolution by modifying the focal length in order to keep the depth error constant, where depth error is affected by quantization noise [6,9]. Variable baseline stereo tracking vision system used to estimate the distance to the object being tracked is described in [19]. This paper contributes to optimizing of constellations in terms of minimizing the depth and matching error, increasing the overlap with adjacent coverage patches, and achieving as high target resolution as the mission objective allows.

Among different coordination aspects, we focus on task assignment in this paper. However, when we introduce a dynamic environment with incomplete knowledge about its behaviour, our static task assignment problem becomes a dynamic decision problem. On top of that, the problem includes two aspects: task decomposition and task allocation. Some of the solutions to task allocation comprise market-based approaches [12,20,17], game theoretical and machine learning approaches [14,23,27], optimization-based approaches [2,5,8], etc. We adopt the market-based approach from [16], due to its simplicity, dynamic response and decentralization. Moreover, we improve the approach by elaborating on the task decomposition aspect in terms of adaptation to the partial available environment knowledge of drones.

## 3 Problem Definition

The simultaneous coverage problem can be formalized as follows: A set of $m$ drones $D = \{d_1, \ldots, d_m\}$ covers a 3D environment which includes a set of ground objects of interest $O = \{o_1, \ldots, o_p\}$, whose position and shape are initially unknown. We consider static objects that neither change their position nor shape, and semi-dynamic objects that don't change their position but may experience some dynamics of their shape, e.g. due to wind. After detection and sufficient observation, an object $o_i$ is abstracted by a set of surfaces $S_i = \{s_{i1}, \ldots, s_{in}\}$. A constellation $c_i$ represents the $k$ viewpoints, i.e. the positions and orientations of the drones required to cover a surface or parts of it. A (part of a) surface is simultaneously covered if $k$ drones visit the viewpoints of the corresponding constellation and capture overlapping images concurrently.

The mission is achieved when the specified area is fully explored and every surface of the detected objects has been simultaneously covered. The overall objective is to complete the mission as fast as possible and at a certain quality.

Solving the simultaneous coverage problem can be decomposed into several interdependent sub-problems: (1) exploring the environment, (2) detecting objects, abstracting surfaces and computing the required constellations, (3) assigning drones to constellations, and (4) covering all surfaces by following collision-free paths between the constellations. In our approach, drones autonomously take on different roles when contributing to the different sub-problems.

## 4 Approach

### 4.1 Mission Objective

Our overall objective is to cover every surface satisfying a certain quality as soon as possible. We have thus two sub-objectives: the mission duration and the coverage quality. We represent the mission duration objective by the number of constellation points $J_{nc}$, since the time it takes a drone to cover each point comprises the time to reach it, to decelerate and stabilize as well as to wait for the other $k-1$ drones in order to simultaneously capture images. The number of constellation points is dependent on the coverage quality: For achieving higher quality, more constellation points are necessary. We label the coverage quality, represented by the resolution $\delta$, with $J_{cq}$, and formulate the mission objective as minimizing

$$J_{mo} = \lambda \cdot J_{nc} + (1 - \lambda) \cdot J_{cq}, \tag{1}$$

where $\lambda \in \mathbb{R} | 0 \leq \lambda \leq 1$ controls the effect of the two optimization goals. We define $J_{cq} = \frac{\delta_{max}}{\delta}$ to be within the limits imposed by simple camera model with image width $w$, minimum safe distance to the surface $D_{max}$, and the lens horizontal aperture angle $\alpha_H$, i.e. $\delta_{max} = \frac{w}{2 \cdot D_{max} \cdot tan(\frac{\alpha_H}{2})}$ [24]. As we aim to minimize this objective, we need to increase $\delta$ to increase the coverage quality.

We estimate the number of constellation points $J_{nc}$ by

$$J_{nc} = \frac{2 \cdot k \cdot \sum_{i=1}^{n} L_i \cdot \lambda_{as} \cdot H_i \cdot \delta^2}{w \cdot h}, \tag{2}$$

which depends on $k$, the resolution $\delta$, image width $w$ and height $h$, a sum $\sum_{i=1}^{n} L_i$ of lengths of all surfaces of an object $o_i$ (double the value due to the requirement of at least 50% horizontal overlap), the object's height $H_i$, and a weight parameter $\lambda_{as} \geq 1$. As opposed to $J_{cq}$, a low value of $\delta$ reduces the mission duration. We use $\lambda_{as}$ throughout the mission to tune the importance of maximizing additional covered area for the purpose of increasing the overlap with adjacent image patches. When $\lambda_{as}$ is set to 1, we obtain no vertical additional area that leads to no overlap between images taken from two neighboring surfaces (in a vertical direction).

### 4.2 Selection of Drone Roles

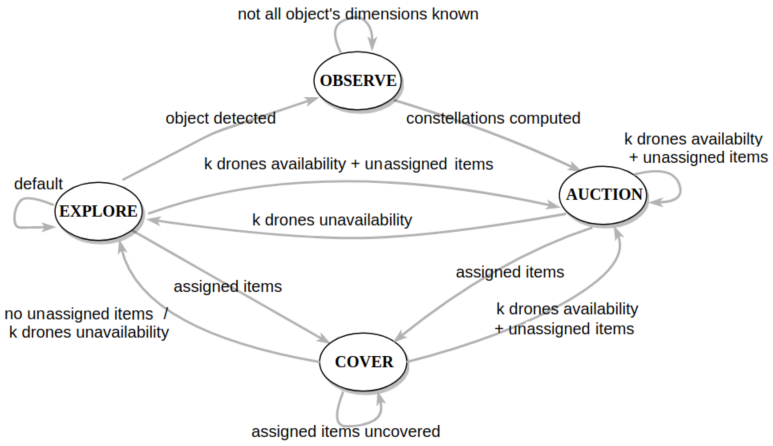



Fig. 2: Four drone roles and transition conditions.

In order to find a solution to the simultaneous coverage problem, we decompose it into smaller sub-problems and adopt a bottom-up approach by solving them individually. These sub-problems comprise exploration, object detection and observation, and coverage, and introduce different roles that drones can take on during the mission. Fig. 2 depicts the drone roles and the predefined transition conditions by means of a state diagram. The default role of a drone $d_i$ is *explore*, represented as $d_i^e$, where it moves in the environment following a given exploration strategy for detecting objects. Once an object has been detected, the drone changes its role to *observe* $d_i^o$, investigates the shape of the object in order to abstract surfaces, and computes constellation points. These points represent the positions and orientations of $k$ cameras satisfying the quality constraints (cp. Sec. 4.3). Afterwards, the computed constellation points to cover the detected object(s) have to be assigned to appropriate drones. Therefore, the drone changes its role to *auction* $d_i^a$ to perform this task assignment by means of different auction strategies (cp. Sec. 4.4). The assigned drones change to the *cover* role $d_i^c$ and execute the necessary steps for covering the assigned surface(s).

Drone assignment is not successful if less than $k$ drones are available for covering an object or surface, respectively. In this case, the drone can either change to *explore* or participate in the bidding of another auction drone. This drone periodically checks if a sufficient number of drones has become available and continues then with the auctioning of unassigned items as long as this condition is satisfied.

Only drones with role $d_i^e$ and role $d_i^a$ but an insufficient number of bidders can participate in the bidding. We further assume that drones with roles $d_i^o$ and $d_i^c$ cannot be interrupted by a bidding request.

### 4.3 Constellation Computation

Fig. 3 sketches a constellation of three drones placed at positions $[x_i, y_i, z_i]$, separated by baselines $b_{ij}$, and at distance $\hat{y}$ to the surface. For simplicity, the covered partition of length $L_p$ is perpendicular to the $y$ axis, and the cameras' views are perpendicular to the surface. The covered partition $P_{sij} \subseteq S_{ij}$ is defined by the points $p_{xyz} = [x, y, z]$ which are visible to all $k$ cameras. If we apply a simple camera model with focal length $f$, sensor dimensions $w \times h$ and aperture angles $\alpha_H$ and $\alpha_V$, the following constraints between any camera pair $i$ and $j$ must hold:

$$\begin{aligned} |x - x_i - \frac{b_{ij}}{2}| &\leq \frac{w\hat{y} - b_{ij}f}{2f}, \\ y - y_i &\geq \frac{b_{ij}f}{w}, \\ |z - z_i| &\leq \frac{\hat{y}h}{2f}. \end{aligned} \tag{3}$$

Note that these constraints impose an overlap in the cameras' FoV of at least 50%. The partition can be specified by the four corner points of $P_{sij}$ as $[x_P, y_P, z_P], [x_P + L_P, y_P, z_P], [x_P + L_P, y_P, z_P + H_P], [x_P, y_P, z_P + H_P]$, where $L_P$ and $H_P$ represent the width and the height of $P_{sij}$, respectively.

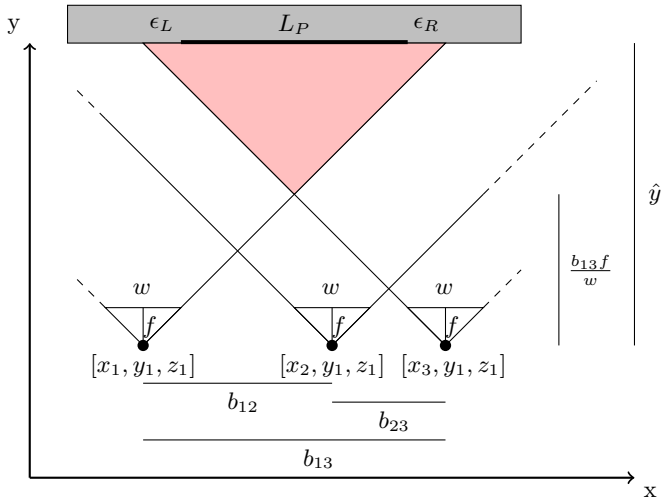



Fig. 3: Drone constellation with covered FoV represented in the $xy$ plane. $k = 3$ drones, positioned at $[x_i, y_1, z_1]$, simultaneously cover the partition of length $L_p$.

In our approach, we formulate the constellation computation problem as a multi-objective optimization problem, having three fitness functions to be minimized:

$$J_{co} = \lambda_{de} \cdot J_{de} + \lambda_{me} \cdot J_{me} - \lambda_{as} \cdot J_{as}, \tag{4}$$

where $J_{de}$ represents the depth error, $J_{me}$ represents the matching error, and $J_{as}$ corresponds to the covered area.

We use a first order Taylor series approximation to estimate the depth error $J_{de} = \frac{\hat{y}^2}{b_{ij} f} \cdot \epsilon_d$ given the distance $\hat{y}$, the disparity error $\epsilon_d$ caused by the corresponding pixels' difference in $x$ coordinates, baseline $b_{ij}$, and the focal length $f$. The matching error $J_{me}$ depends on the parameters $a$ and $b$ describing matching performance and the relative viewing angle $\gamma$, and is approximated as $\frac{1}{a} \cdot \ (e^{-b|\gamma|} - 1)$ [18,24].

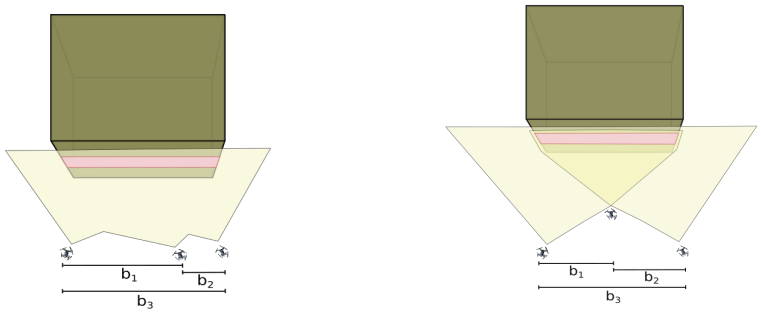
The third cost function component $J_{as}$ rewards the additional area covered beyond the border of the partition $P_{sij}$. This additional overlap with adjacent image patches helps to improve the overall stitching of the images from the individual partitions. Basically, we want to increase both $\epsilon_R$ and $\epsilon_L$ in Fig. 3 and therefore subtract it in Eq. 4, as opposed to the other objectives.

With the equal height of the $k$ constellation points, the $y$ and $z$ coordinates of the drones are given as $y_i = y_P - \frac{L_P}{tan(\frac{\alpha_H}{2})}$ and $z_i = z_P + \frac{H_P}{2}$. Note that if $k = 1$ only $J_{as}$ is taken into account aiming for an overlap with neighboring partitions of at least 50%.

Since constellation computation is an NP-hard multi-objective optimization problem, we address it with an efficient evolutionary algorithm NSGA-II (non-dominated sorting genetic algorithm-II), leading to Pareto optimal solutions [7]. This fast, elitist and parameterless algorithm is known for its low computational complexity with a simple but efficient constraint-handling method and fast non-dominated sorting procedure resulting in improved convergence. We limit the number of function evaluations of the algorithm based on the mission's update rate.

As previously stated, $k$ controls the required simultaneously captured images for a surface. Thus, a larger $k$ may provide more data about dynamic textures, occlusions, etc., which might be beneficial for certain applications. Therefore, we extend our constellation computation algorithm to be suitable for a larger $k$. Fig. 4 depicts two relaxations for the constellation computation for $k = 3$ drones. We allow (a) asymmetric drone placements resulting in different baseline settings and (b) different distances between drones and the covered surface resulting in different target resolutions.

Since we consider 3D objects in the environment, we have to assure the coverage of all visible surfaces, including the top ones. We do so by projecting the top surface of the object as an additional vertical partition to the vertical surface being covered. However, we are aware of a lower achievable resolution due to the fact that the far most part of the top surface is further away than the vertical surface. To assure corner detection in the images, we add another horizontal partition to the right of each surface.




(a) (b)

Fig. 4: Different constellation relaxations for $k = 3$: (a) different baselines, and (b) different distances to the surface.

### 4.4 Adaptive Task Assignment

After constellation computation has been completed, drone $d_i$ changes its role to $d_i^a$ and starts the assignment of available drones to constellations. We adopt a decentralized market-based assignment due to its scalability, efficiency, and suitability for dynamic settings [1]. Optimal assignment algorithms that reach the global optimum are mostly centralized. Many decentralized multi-agent task assignment methods employ auctioning mechanisms which have been applied to NP-hard problems (routing, scheduling, planning, etc.), or address problems where only partial knowledge about the environment is available, and therefore local information is exploited. Since our mission is defined in such a way, agents make decisions based on their current (local) knowledge. Due to the lack of knowledge, most often the global optimum solution is not found. When, on top of that, we consider a dynamic environment and only agents' local knowledge about it, the resulting assignment quality gets difficult to measure.

We adopt our previous approach [16], where we deployed a basic auction mechanism enabling drones to bid for constellations and introduced an object-based (OB) and surface-based (SB) variant. OB allocation assigns all constellation points of an object to the same team of $k$ drones, whereas SB allocation assigns constellations individually to $k$ drones.

Algorithm 1 describes our new adaptive allocation algorithm which is running on each drone $d_i$ with unassigned constellations. As long as drone $d_i$ has unassigned allocation items, it is free to explore around and periodically (line 7) broadcast the request for roles in order to find out the number of drones available for bidding (lines 8 to 11). If there are at least $k$ drones available, $d_i$ selects an object from $O_i^a$, computes the constellations of object $o$ and initiates an auction

**Algorithm 1** Market-based Constellation Assignment

**Input:** set of unassigned objects $O_i^a$ of drone $d_i$, set of surfaces $S_i$ for $o_i \in O_i^a$, auction $T_a$ and request $T_r$ timeout

1: $F_i^a \leftarrow$ set of drones free to bid to $d_i$
2: $B_i^a \leftarrow$ set of drones that bid to $d_i$
3: $C_i^a \leftarrow$ set of constellation points to assign
4: $S_i^a \leftarrow$ subset of $S_i$ being assigned to winners in $B_i^a$
5: $t_a = 0, t_r = T_r \leftarrow$ auction and request timeout counter
6: **while** $O_i^a \neq \emptyset$ **do**
7: **if** $t_r \geq T_r$ **then**
8: $d_i \leftarrow d_i^e$
9: $broadcast\_role\_request()$
10: $t_r \leftarrow 0$
11: $F_i^a \leftarrow update\_free\_drones()$
12: **if** $|F_i^a| \geq k$ **then**
13: $d_i \leftarrow d_i^a$
14: $o \leftarrow select\_object(O_i^a)$
15: $C_i^a \leftarrow compute\_constellations(o)$
16: $broadcast\_constellations(C_i^a)$
17: **if** $t_a < T_a$ **then**
18: $t_a \leftarrow t_a + 1$
19: $B_i^a \leftarrow update\_bids()$
20: **if** $|B_i^a| \geq k$ **then**
21: $S_i^a \leftarrow allocate\_surfaces()$
22: $broadcast\_decision(S_i^a)$
23: $wait\_for\_acknowledgment()$
24: $O_i^a \leftarrow O_i^a \setminus o$; $t_r \leftarrow T_r$
25: **break**
26: **else**
27: $d_i \leftarrow d_i^e$
28: $t_r \leftarrow T_r$; $t_a \leftarrow 0$
29: **else**
30: $t_r \leftarrow t_r + 1$
31: $d_i \leftarrow d_i^e$

by broadcasting the constellations of $o$ (lines 14 to 16). It does so as long as it has not received enough bids (lines 19 and 20) and the timeout $t_a$ has not expired. If a sufficient number of bids has been received, $d_i$ selects the $k$ drones from $B_i^a$ with the highest bid values and broadcasts a decision (lines 21 and 22).

We consider this part to be of importance when adapting to the dynamic change of roles and the current knowledge drones possess. This knowledge comprises a number of drones, their roles and locations, and the shape of the object and its constellations. Basically, we improve the task decomposition aspect of the task assignment problem by assigning a number of neighboring surfaces proportionally to the number of bids and their distances to the drones that bid. This way we parallelize the coverage when multiples of $k$ drones bid. This is im-

portant because with such a dynamic environment and quick changes, especially in drones' movements, proceeding with an assumption made at the moment an object was ready to be assigned could prolong the mission.

Once the drone $d_i$ receives an acknowledgment from all allocated drones (line 23), it removes the object from $O_i^a$ and resets timer $t_r$ to be ready to broadcast role requests again (line 24). If the timeout has exceeded, $d_i$ keeps exploring and resets both timers to broadcast role requests again (lines 27 and 28). If all objects have been assigned, $d_i$ changes its role to explore (line 31).

To consider the auction successful, drone $d_i$ waits for the acknowledgments from all assigned drones. This acknowledgment mechanism also holds for the bidding drones. If they did not receive an acknowledgment from the drone $d_i$ within a predefined timeout $T_c$, they cancel the bid and start exploring or submit a bid to another auction.

As stated above, due to the partial environment knowledge, we cannot aim for the global optimum solution, rather for the optimal solution given the knowledge a drone auctioneer possesses at the moment it starts the auction. One reason for a non-optimal outcome could be an unknown obstacle between assigned drones(s) and constellation points, which results in a prolonged flight time as compared to the estimated time of the drone bidder.

### 4.5 Exploration and Object Detection

We apply a simple heuristic for exploring the environment to detect objects of interest. The drones start exploring from the ground station in a random direction and move straight until they detect an object, encounter an obstacle, or reach the border of the environment. In the latter case, they rotate at a random angle to stay within the environment and continue exploration. For object and obstacle detection we exploit the drone's frontal camera. In particular, we leverage the API of AirSim to retrieve an uncompressed depth image from the left frontal camera, convert it to a gray-scale image, and remove the ground to show only the relevant part. We estimate the distance to a potential object by evaluating the corresponding pixel values and perform further analysis based on the size of the pixel cluster in the depth image. To investigate the shape of an object, the drone performs wall following by keeping the distance to the object fixed while moving around the object. Once wall following has reached the starting position, i.e. the path around the object has been closed, the drone ascends to estimate the height of the object.

We adapt the path planning approach [16] by considering online obstacle detection and avoidance. Whenever an object or obstacle is detected on the computed path towards a constellation point, we perform a similar wall following approach for bypassing the obstacle at a safe distance. If the path towards the constellation point is clear, the drone continues in a straight line.

# 5 Experiments

## 5.1 Experimental Setup

We use the open-source simulator Microsoft AirSim[1] (Aerial Informatics and Robotics Simulation) which is built on the Epic Games' Unreal Engine 4 (UE4)[2] for our simulation study. We created a virtual environment on a 64-bit Windows-10 platform (cp. Fig. 5), imported it to a Linux platform running Ubuntu 16.04 LTS, and added the AirSim plugin as a replacement for AirSim drone with the *SimpleFlight* built-in flight controller. As we aim for a fully decentralized system and want to deploy our algorithms on real drones, we incorporated the recent ROS2 version (distribution Bouncy Bolson)[3]. With this setup the drone agents are able to exchange messages at an update rate of 1 $s$ which is very important for real-world missions. ROS2 is supposed to run under Windows. However, in our 64-bit Windows-10 setup, ROS2 showed unreliable performance, in particular, more than 50% of the messages were lost when running simulations with more than 4 drones. Therefore, we used the Windows platform only for creating the environment and the Linux platform for running the experiments and exchanging messages among the drone agents via ROS2.

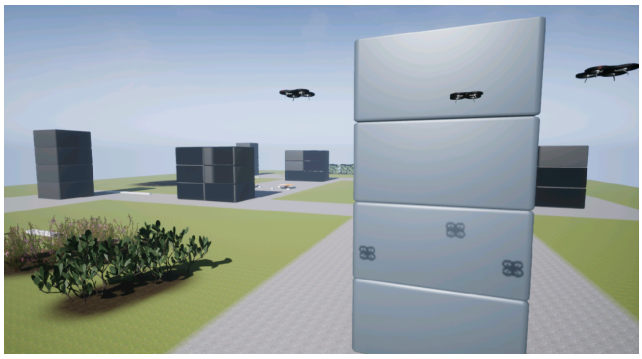
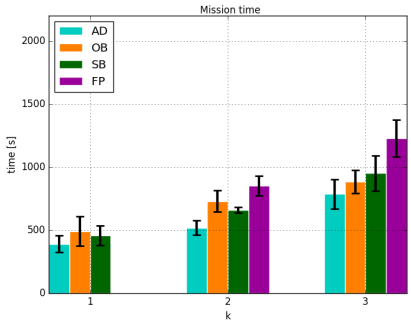


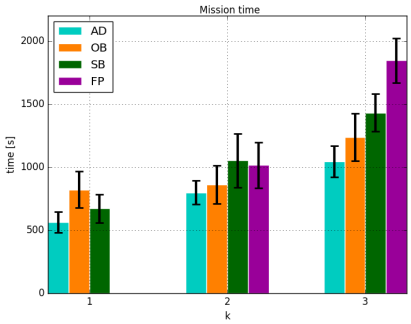
Fig. 5: The virtual environment for our simulation study rendered by the Epic Games Unreal Engine 4.

For our simulation we use an environment of 240 $m \times 240$ $m$ and place building blocks of different sizes as objects of interest. The ground station is at the center from where all drones start their mission. The drones are equipped with cameras, GPS, an Inertial Measurement Unit (IMU), and a barometer. We perform experiments with varying $k \in \{1, 2, 3\}$ and for the object-based (OB), surface-based (SB), and adaptive (AD) auction variants. As baseline for

[1] https://github.com/Microsoft/AirSim
[2] https://www.unrealengine.com
[3] https://github.com/ros2/ros2




(a) Scenario A




(b) Scenario B

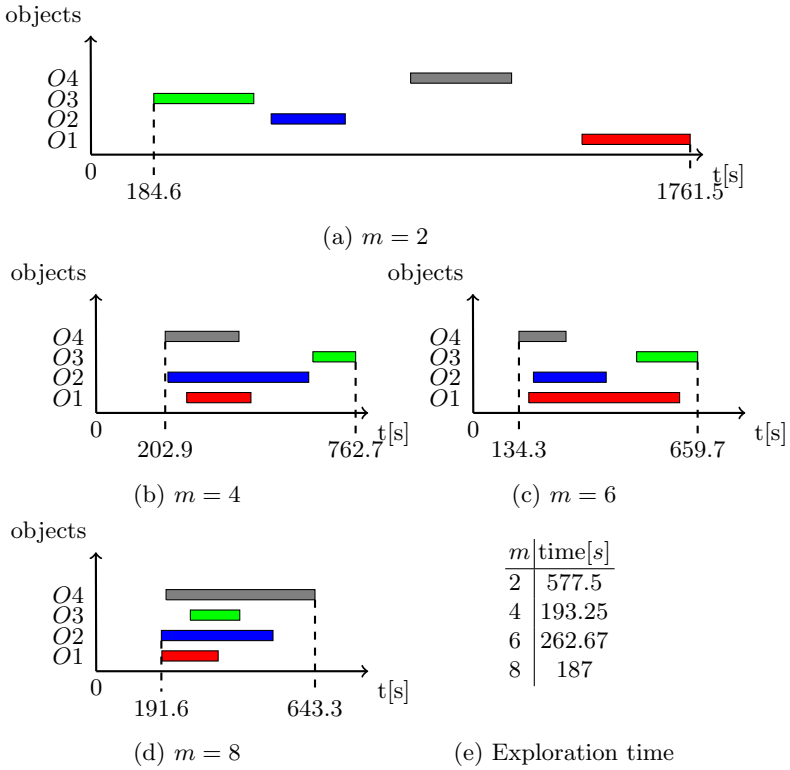
| | $\delta[\frac{px}{m}]$ |
|---|---|
| AD | $62.53 \pm 8.32$ |
| OB | 50 |
| SB | 50 |
| FP | 50 |

(c) Achieved $\delta$

Fig. 6: Average mission time with standard deviations for adaptive auction (AD), object-based (OB), surface-based (SB), and fixed pair (FP) assignment for scenario A (6a) and scenario B (6b). Achieved pixel resolution $\delta$ (6c).

the comparison we also run experiments with fixed teams of $k$ drones (FP), i.e. offline assigned teams jointly explore the environment and simultaneously cover the detected surfaces and no dynamic assignment is necessary.

We measure the total mission time and the achieved pixel resolution $\delta$ as key performance metrics. We further measure the object coverage time, which is defined as the time period when an object has been detected until it is fully covered, as well as the times the drones operate in the four different roles. We compare our approach with two different settings: Scenario A is composed of 4, and Scenario B is composed of 6 unevenly distributed building blocks. We run the experiments with varying number of drones $m \in \{2, 4, 6, 8\}$. Important simulation parameters are fixed as follows: The timeout thresholds $T_r$, $T_a$ and $T_c$ are set to 3 $s$, 5 $s$ and 10 $s$, respectively. We use the following camera parameters: $f = 1662.8\ px$, $\alpha_H = 60°$, $w \times h = 1920 \times 1080\ px$ (full HD sensor). In order to give more priority to the mission's duration over coverage quality, we set $\lambda$ to 0.7, whereas we set $\lambda_{as}$ to be equal to 1.5 to ensure 50% of the vertical overlap. Both minimum distance between drones and a distance to objects are set to 3 $m$. We run 10 simulations for each experiment in order to lower the effect of randomness in exploration. Since we consider the mission to be successfully completed when the whole area is known and all objects have been covered, we terminate the experiments when all objects have been completely covered, assuming we know a priori the number of objects.




(a) $m = 2$

(b) $m = 4$

(c) $m = 6$

(d) $m = 8$

| $m$ | time$[s]$ |
|---|---|
| 2 | 577.5 |
| 4 | 193.25 |
| 6 | 262.67 |
| 8 | 187 |

(e) Exploration time

Fig. 7: Impact of number of drones $m$ on time each object required to be fully covered from the moment it was ready to be assigned, and the average exploration time per drone (7e), for the environment setup from Scenario A.

### 5.2 Experimental Results

Fig. 6 depicts the impact of $k$ and the assignment variant (AD, OB, SB, and FP) on the overall mission time for both scenarios. It is rather intuitive that the coverage time increases with $k$ as more drones have to be jointly assigned and potentially wait for each other at the assigned points. In the AD auction variant, we aim to take advantage of both the OB and the SB variant, and on top of that, adapt the coverage quality to mission requirements. As shown in Fig. 6a for $k = 1$ and $k = 2$ and in Fig. 6b for $k = 1$, the SB variant outperforms the OB variant due to a sufficiently large number of drones $m$. If the number of objects and hence unassigned tasks is relatively small as compared to the available drones, the OB variant outperforms the SB variant, since the auctioneering drones do not lose time on waiting for bids for each surface. In this case, assignment of drones for the whole object is more efficient.

Our AD variant outperforms the other variants since it adapts to the mission and its dynamics of discovering new objects. Furthermore, it achieves a higher

pixel resolution $\delta$ as the fixed one of the other three variants (cp. Fig. 6c). Regarding a comparison between fixed and dynamic assignment, we analyze how the FP variant performs with varying $k$ in our scenarios. For $k = 1$, the FP variant performs as the OB variant, since the same drone which has detected the object can also cover it. For $k = 2$, FP performs only slightly worse than OB or SB. For $k = 3$, the performance of FP deteriorates. The main reason for this degraded performance is that the FP variant explores the environment in fixed teams of $k$ co-located drones and is therefore less effective in detecting objects as compared to the exploration with independent drones. Even though the FP variant's advantages in terms of shorter coverage time are not evident in the total mission time, drones did cover some objects faster for certain scenarios because of the proximity of the other $k - 1$ drone(s). Moreover, we have introduced two relaxations for $k = 3$: different baselines and different distances to the surface. From the coverage time perspective, the effect of these relaxations is negligible because the distance variation of a $k$ constellation point for two relaxations is much smaller than the distances between the constellation points. However, the coverage quality can benefit from these relaxations, and we therefore applied both relaxations for our simulations with $k = 3$, i.e. variable distances at corners to increase the overlap and variable baselines for regular surfaces.

We further evaluated the scalability of our AD variant by varying $m \in \{2, 4, 6, 8\}$ for scenario A with $k = 2$. The horizontal bars in Fig. 7 show the object coverage time for all objects in order to visualize the overall mission execution and the effect of exploration for different $m$. The left value on the $x$ axis represents the time when the first object has been detected, whereas the right value represents the overall mission time. For $m = k = 2$, we can clearly observe the sequential coverage of the four objects (Fig. 7a); the gaps in between correspond to the time required for detection and abstraction of surfaces. Figures 7b to 7d plot results for $m > k$, where objects can be covered in parallel. Note that coverage of a particular object can be interrupted due to too few drones available. Fig. 7e summarizes the time drones explored the environment searching for objects. This exploration time does not decrease with increasing number of drones. It strongly depends on the uncertainty of the environment (i.e. random object placement) and chosen exploration method.

## 6 Conclusions

We have presented a decentralized coordination method to simultaneously cover a priori unknown environments aiming for minimizing the mission duration and maximizing the coverage quality. The allocation of drones to constellation points is a critical step for this problem. Our adaptive market-based assignment (AD) achieved a shorter mission duration as compared to surface-based (SB), object-based (OB) or fixed assignments (FP) in our simulation study, as well as a higher coverage quality. Since we apply our approach in initially unknown environments, the time required for exploration has a significant influence on the overall mission

time. Thus, there is a tradeoff between exploration and coverage and the effort put into (concurrently) solving these sub-problems.

Simultaneous coverage represents a challenging multi-agent problem, and efficient solutions will leverage various applications including monitoring, inspection and reconstruction. As future work we intend to investigate in (i) coordinated exploration methods to decrease the object detection time, (ii) considering the object's semantics to adapt $k$ and $\delta$ for each object individually, and (iii) in deploying our dynamic coordination in real multi-drone applications.

**Acknowledgments.** This work is supported by the Karl Popper Kolleg on Networked Autonomous Aerial Vehicles (uav.aau.at) at the University of Klagenfurt.

## References

1. Badreldin, M., Hussein, A., Khamis, A.: A comparative study between optimization and market-based approaches to multi-robot task allocation. Advances in Artificial Intelligence **2013**, 12 (2013). https://doi.org/10.1155/2013/256524
2. Berman, S., Halász, Á., Hsieh, M.A., Kumar, V.: Optimized stochastic policies for task allocation in swarms of robots. IEEE Transactions on Robotics **25**(4), 927–937 (2009). https://doi.org/10.1109/TRO.2009.2024997
3. Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R.: Receding horizon "next-best-view" planner for 3D exploration. In: Proc. IEEE Int'l Conference on Robotics and Automation. pp. 1462–1468. IEEE (2016). https://doi.org/10.1109/ICRA.2016.7487281
4. Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R.: Receding horizon path planning for 3D exploration and surface inspection. Autonomous Robots **42**(2), 291–306 (2018). https://doi.org/10.1007/s10514-016-9610-0
5. Capitan, J., Spaan, M.T., Merino, L., Ollero, A.: Decentralized Multi-Robot Cooperation with Auctioned POMDPs. The Int'l Journal of Robotics Research **32**(6), 650–671 (2013). https://doi.org/10.1109/ICRA.2012.6224917
6. Chang, C., Chatterjee, S.: Quantization error analysis in stereo vision. In: Proc. Asilomar Conference on Signals, Systems & Computers. pp. 1037–1041. IEEE (1992). https://doi.org/10.1109/ACSSC.1992.269140
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
8. Fonooni, B., Jevtić, A., Hellström, T., Janlert, L.E.: Applying ant colony optimization algorithms for high-level behavior learning and reproduction from demonstrations. Robotics and Autonomous Systems **65**, 24–39 (2015). https://doi.org/10.1016/j.robot.2014.12.001
9. Freundlich, C., Zhang, Y., Zhu, A.Z., Mordohai, P., Zavlanos, M.M.: Controlling a robotic stereo camera under image quantization noise. The Int'l Journal of Robotics Research **36**(12), 1268–1285 (2017). https://doi.org/10.1177/0278364917735163
10. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robotics and Autonomous systems **61**(12), 1258–1276 (2013). https://doi.org/10.1016/j.robot.2013.09.004
11. Gallup, D., Frahm, J.M., Mordohai, P., Pollefeys, M.: Variable Baseline/Resolution Stereo. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008). https://doi.org/10.1109/CVPR.2008.4587671

12. Gerkey, B.P., Mataric, M.J.: Sold!: Auction Methods for Multirobot Coordination. IEEE Transactions on Robotics and Automation **18**(5), 758–768 (2002)
13. Heng, L., Gotovos, A., Krause, A., Pollefeys, M.: Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In: Proc. IEEE Int'l Conference on Robotics and Automation. pp. 1071–1078. IEEE (2015). https://doi.org/10.1109/ICRA.2015.7139309
14. Jiang, A.X., Procaccia, A.D., Qian, Y., Shah, N., Tambe, M.: Defender (mis) coordination in security games. In: Twenty-Third International Joint Conference on Artificial Intelligence (2013)
15. Khan, A., Rinner, B., Cavallaro, A.: Cooperative Robots to Observe Moving Targets: A Review. IEEE Transactions on Cybernetics **48**(1), 187–198 (2018). https://doi.org/10.1109/TCYB.2016.2628161
16. Mazdin, P., Rinner, B.: Efficient and QoS-Aware Drone Coordination for Simultaneous Environment Coverage. In: Proc. IEEE Conference on Multimedia Information Processing and Retrieval. pp. 333–338. IEEE (2019). https://doi.org/10.1109/MIPR.2019.00066
17. McIntire, M., Nunes, E., Gini, M.: Iterated multi-robot auctions for precedence-constrained task scheduling. In: Proc. Int'l Conference on Autonomous Agents & Multiagent Systems. pp. 1078–1086. Int'l Foundation for Autonomous Agents and Multiagent Systems (2016)
18. Morel, J.M., Yu, G.: ASIFT: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences **2**(2), 438–469 (2009). https://doi.org/10.1137/080732730
19. Nakabo, Y., Mukai, T., Hattori, Y., Takeuchi, Y., Ohnishi, N.: Variable baseline stereo tracking vision system using high-speed linear slider. In: Proc. IEEE Int'l Conference on Robotics and Automation. pp. 1567–1572. IEEE (2005). https://doi.org/10.1109/ROBOT.2005.1570337
20. Nunes, E., Gini, M.: Multi-robot auctions for allocation of tasks with temporal constraints. In: Proc. AAAI Conference on Artificial Intelligence (2015)
21. Perez-Carabaza, S., Scherer, J., Rinner, B., Lopez-Orozco, J.A., Besada-Portas, E.: UAV Trajectory Optimization for Minimum Time Search with Communication Constraints and Collision Avoidance. Engineering Applications of Artificial Intelligence **85**, 357–371 (2019)
22. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Proc. Field and Service Robotics (2017)
23. Tambe, M.: Security and game theory: algorithms, deployed systems, lessons learned. Cambridge university press (2011). https://doi.org/10.1017/CBO9780511973031
24. Weyrer, M., Rinner, B.: UAV Motion Planning and Control for Multi-Coverage of 3D Environments. In: Proc. Int'l Conference on Unmanned Aircraft Systems. pp. 939–946 (2018). https://doi.org/10.1109/ICUAS.2018.8453427
25. Yanmaz, E., Yahyanejad, S., Rinner, B., Hellwagner, H., Bettstetter, C.: Drone Networks: Communications, Coordination, and Sensing. Ad Hoc Networks **68**(1), 1–15 (2018). https://doi.org/10.1016/j.adhoc.2017.09.001
26. Yoder, L., Scherer, S.: Autonomous exploration for infrastructure modeling with a micro aerial vehicle. In: Proc. Field and Service Robotics. pp. 427–440. Springer (2016). https://doi.org/10.1007/978-3-319-27702-8_28
27. Zhang, C., Lesser, V.: Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In: Proc. AAAI Conference on Artificial Intelligence (2011)