Auton Robot (2017) 41:1609–1628
DOI 10.1007/s10514-017-9631-3

Simultaneous area partitioning and allocation for complete coverage by multiple autonomous industrial robots

Mahdi Hassan[1] · **Dikai Liu**[1]

Received: 1 March 2016 / Accepted: 1 April 2017 / Published online: 13 April 2017

Abstract For tasks that require complete coverage of surfaces by multiple autonomous industrial robots, it is important that the robots collaborate to appropriately partition and allocate the surface areas amongst themselves such that the robot team's objectives are optimized. An approach to this problem is presented, which takes into account unstructured and complex 3D environments, and robots with different capabilities. The proposed area partitioning and allocation approach utilizes Voronoi partitioning to partition objects' surfaces, and multi-objective optimization to allocate the partitioned areas to the robots whilst optimizing robot team's objectives. In addition to minimizing the overall completion time and achieving complete coverage, which are objectives particularly useful for applications such as surface cleaning, manipulability measure and joint's torque are also optimized so as to help autonomous industrial robots to operate better in applications such as spray painting and grit-blasting. The approach is validated using six case studies that consist of comparative studies, complex simulated scenarios as well as real scenarios using data obtained from real objects and applications.

Keywords Multi-robot collaboration · Multiple autonomous industrial robots · Area partitioning and allocation · Complete coverage

Electronic supplementary material The online version of this article (doi:10.1007/s10514-017-9631-3) contains supplementary material, which is available to authorized users.

✉ Mahdi Hassan
mahdi.hassan@student.uts.edu.au

Dikai Liu
dikai.liu@uts.edu.au

[1] Centre for Autonomous Systems (CAS), University of Technology Sydney (UTS), Sydney, Australia

1 Introduction

Advancements in Autonomous Industrial Robots (AIRs), through technologies that allow AIRs to be deployed in industrial applications that are dirty or hazardous and environments that are unstructured, can make noticeable improvements in the capacity of numerous manufacturing applications. Thus, through advancements in AIRs, the aim is to aid the AIRs to perform various tasks independently. Examples include performing exploration of an environment for mapping (Paul et al. 2011) and robots' localization within an environment (Carlone et al. 2011), path planning and collision-free motion planning (Guanghui et al. 2012; Latombe 2012), and base placement optimization (Hassan et al. 2016).

By deploying multiple AIRs, a wider range of manufacturing applications is possible to be carried out by the AIRs, and the capacity of the AIRs to perform a task can be increased. As a result, better outcomes in terms of efficiency, productivity and object manipulability can be obtained. However, effective collaboration amongst the AIRs is a crucial requirement for obtaining such outcomes.

As part of the multi-robot collaboration, the AIRs are required to perform a number of tasks specific to the application. For example, if three AIRs such as those shown in Fig. 1 are deployed to perform grit-blasting or spray painting in a complex and unstructured environment, then each AIR is expected to have enough intelligence to perform the following tasks: (1) obtaining a partial map of the environment by exploring and localizing within the environment, (2) sharing the partial map with other AIRs in order for all AIRs to have a complete map of the environment, (3) sharing other information such as AIR's operation status and capabilities with other AIRs, (4) using the shared information to determine an appropriate base position and orientation since the AIR base

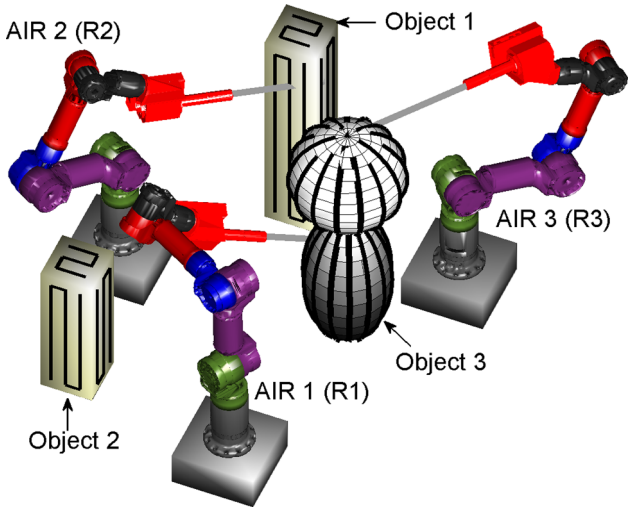


Fig. 1 Three AIRs are used to perform grit-blasting or spray painting on three different objects which are separated from each other

is assumed to be fixed during the task execution, (5) achieving complete coverage of the surfaces under consideration, and (6) task execution. Considering optimizing team's objectives throughout such tasks is important for optimal operation and collaboration of the AIRs during the task execution.

Manufacturing applications that require complete surface coverage such as surface cleaning, grit-blasting and spray painting can make great use of multiple AIRs. These kind of applications are the main focus of this paper. An important task in such applications is complete coverage, i.e. all surface areas of interest need to be operated on by the AIRs. This paper focuses on collaborative complete coverage by presenting an approach for simultaneously partitioning the surface areas of interest (or multiple unconnected surface areas) and appropriately allocating the partitioned areas to each of the deployed AIRs. The approach is carried out such that ultimately each AIR is assigned a fair portion of the surfaces whilst AIR team's objectives are optimized. A single robot Coverage Path Planning (CPP) algorithm (Galceran and Carreras 2013) can then be implemented on the allocated areas of each AIR. Selection of the appropriate CPP algorithm can be based on the environment (e.g. planar, rectilinear, polygonal, 3D, convex/nonconvex, etc.) or the specific application under consideration; this is outside the scope of the paper. The approach is tailored to unstructured environments and real-life operation where the point cloud generated and processed after the exploration and mapping phase can be directly used and the complex task of acquiring a 3D model such as a mesh is not necessary. The deployed AIRs can have different capabilities such as different speed and end-effector tool size. Unlike simple mobile robots such as those used for floor cleaning, in order to obtain appropriate solutions in manufacturing applications such as grit-blasting and spray painting where the AIRs are equipped with manipulators, the performance and the capabilities of the manipulators should be taken into account. Therefore, in addition to considering the objectives of minimizing the overall completion time and maximizing coverage, torque minimization and manipulability measure maximization are also taken into account for the better operation of the manipulators during the task execution.

This work is a continuation of the work in Hassan et al. (2014) where a simple approach was developed for 2D and planar objects. The extension of the work includes: (1) an approach that can effectively partition and allocate 3D and non-planar objects/surfaces amongst multiple AIRs, (2) testing of the approach for multiple objects that are separated from each other, such as those shown in Fig. 1, (3) incorporating manipulability measure to the AIR team's objectives and analyzing the effect of it, (4) case studies based on data obtained from real applications for the validation of the approach, and (5) comparisons with a pattern-based genetic algorithm approach.

The rest of the paper is organized as follows. Section 2 reviews some of the algorithms related to the complete coverage problem. Section 3 presents a detailed explanation of the problem and the requirements that need to be satisfied as part of the solution to the problem. The proposed approach is presented in Sect. 4 and the mathematical model is detailed in Sect. 5. Case studies are presented in Sect. 6 to demonstrate the effectiveness of the approach for different scenarios and conditions. Discussions on limitations is then provided in Sect. 7. Finally, Sect. 8 concludes the presented work and suggests future work.

2 Related work

There are numerous algorithms available for a single robot to perform a complete coverage task and effectively cover the surfaces of interest online or offline. Galceran and Carreras (2013) provide a survey on the recently developed algorithms, and divide the algorithms into categories such as classical exact cellular decomposition, Morse-based cellular decomposition, landmark-based topological coverage, contact sensor-based coverage of rectilinear environments, grid-based coverage, and graph-based coverage. Recently, research is being focused on multi-robot coverage for different environments and applications.

Zheng et al. (2010) deal with the problem of terrain coverage by multiple robots. Their algorithm can handle a terrain with non-uniform traversability, i.e. a terrain that does not have a constant traversing time at all locations and hence, the terrain is weighted. This algorithm extends the Multi-robot Forest Coverage (MFC) algorithm to be compatible with a terrain that has non-uniform traversability. Compar-

isons are made with the Multi-robot Spanning Tree Coverage (MSTC), which falls in the grid-based coverage category, to demonstrate that the algorithm can provide solutions with smaller coverage time.

Several bio-inspired multi-robot coverage approaches have been developed, which are based on the behaviors found in nature (Galceran and Carreras 2013). A recent example is presented by Ranjbar-Sahraei et al. (2012) where the behavior of ants is used to disallow other robots to enter the boundaries of a robot. In this work, the robots move in a circular motion in a 2D environment, and similar to ants, they use the strategy of depositing pheromones on the border of their territories to prevent or reduce intersection of borders. That is, if any robot detects pheromone, it changes its direction of circular motion and hence avoids intersection with another robot's border. As a result, the robots gradually spread out in the environment. The paper also presents two extensions to the approach. In the first extension, the radius of the circular motion of the robots is increased if the likelihood of detecting pheromone is small, and vice versa. The second extension allows for behavior change when an intruder is detected by decreasing the territory areas of the robots.

An algorithm that falls in the category of sensor-based coverage of rectilinear environments is presented in Kapanoglu et al. (2012). In this work, the environment is represented as a set of overlapping disks. The mobile robots pass through the center of the disks to cover the environment. To determine the visiting order of the disks by a robot, the initial location of the robots is considered, and a set of patterns from 8 predefined patterns are chosen. Genetic Algorithm (GA) is used, and for each robot, the priority index of the robot, the index of the selected patterns, and the number of moves using each pattern are decided. Each chromosome contains these parameters for all robots, and the parameters are organized in the chromosomes with the aim of reducing the search space. The fitness value is designed to be the makespan or the overall completion time. The objective is therefore to minimize the makespan.

The work in Janchiv et al. (2013) uses exact cellular decomposition method and flow networks to achieve complete coverage using a single or multiple mobile robots. In this work, using a sweeping line, the free space of the environment is first broken down into a number of cells. The cells which form the nodes of the flow network are approximated to be rectangular or trapezoidal. A search algorithm is then used to find a minimum cost path from the flow network. Twelve different templates are developed such that each cell can be covered using the back-and-forth motion specified in the templates. The total coverage time of a robot is the sum of the times to cover the cells and the times to move between the cells in the order selected from the flow network.

Fazli et al. (2013) developed several algorithms for the problem of multi-robot repeated area coverage. In brief, the overall approach is as follows: (1) first generating a number of points, called static guards, on the environment such that the whole environment can be jointly observed if it is assumed that there are as many robots as there are guards, (2) creating a graph by appropriately joining the guards and the nodes of the workspace, (3) reducing the size of the graph, and (4) covering the graph using either the cluster-based coverage (where the graph is divided into n clusters, n being the number of robots) or cyclic coverage (where the aim is to find the shortest tour by going through all static guards and then appropriately allocating a portion of the tour to each robot). For each of the stages mentioned above, the author presents one or more algorithms to tackle the individual problems.

There are many studies dedicated to multi-robot coverage for aerial vehicles (Galceran and Carreras 2013, p. 1272). Many researchers have used or studied convex decomposition (Ren et al. 2013) to divide an arbitrary shape into a number of convex or near-convex polygons. Maza and Ollero (2007) have used convex decomposition to divide the search environment. They present algorithms that enable multiple heterogeneous Unmanned Aerial Vehicles (UAVs) to cooperative with each other to find an object in the search environment. Each robot is then responsible for covering a convex polygon using a zigzag pattern.

The work in Gunady et al. (2014) first performs partitioning of the area of interest, which is referred to as multi-seeker's territory division in the paper. The approach is designed to tackle the problem of Hide-and-Seek present in many applications such as disaster rescuing and mine detection. It presents a hierarchical scheme for Reinforcement Learning (RL) used to solve the problem. Multiple levels are utilized in the scheme. As an example, in the mine detection application, the lower level deals with reaching a candidate mine location from the current location, whereas the higher level deals with generating a trajectory for each robot to obtain a full scan.

Danner and Kavraki (2000) presented a sampling-based approach to the problem of inspecting confined spaces. The idea is to find a number of points on the environment such that by visiting these points through a calculated short path the entire boundary can be inspected. In recent years, significant progress has been made on probabilistic sampling-based algorithms which are proven to be effective for handling complex environments (Galceran and Carreras 2013). The works in Englot and Hover (2012), Englot and Hover (2013) present a sampling-based approach for inspection of complex and visually occluded environments such as ship hulls and marine structures where obstacles are present. In the papers, probabilistic completeness of the sampling-based coverage algorithm is demonstrated, an iterative algorithm that aims to improve the sampling-based coverage path is presented, and experimental results are included to show the remarkable

efficiency of the approach. It will be interesting to extend the approach to multi-robot inspection.

Unlike most of the above work, the presented approach focuses on simultaneous area partitioning and allocation rather than Coverage Path Planning (CPP). It has the flexibility to then enable a suitable existing single robot CPP algorithm to be applied to the allocated area of each AIR based on the shape of the object and the application under consideration. In many of the existing methods, e.g. Ranjbar-Sahraei et al. (2012), Kapanoglu et al. (2012), Janchiv et al. (2013), and Fazli et al. (2013), the developed algorithms are shown to be efficient for planar or projectively planar (2.5D) surfaces, whereas the presented approach considers complete coverage for applications that require the AIRs to operate on 3D objects that may be non-planar, complex in shape and separated (unconnected) from each other. Some of the algorithms are designed for specific environments; for example in Kapanoglu et al. (2012) the approach has only been validated using rectilinear environments and rectilinear moves (90° and 180° robot turns). The proposed approach does not have these limitations. Additionally, the proposed approach has the advantage of being able to use the point cloud information that is generated from sensing the environment. This aspect is particularly helpful since in real-life scenarios, exploration and mapping of unstructured environments may be necessary and hence, acquiring a 3D model to generate a mesh can be much more difficult than directly using the point cloud. Furthermore, in addition to considering the general objectives of minimizing the overall completion time and achieving complete coverage, the performance and the capabilities of the manipulators are taken into account to optimize torque and manipulability.

3 Problem definition

As an example scenario, consider the environment in Fig. 2a where the I-beam needs to be grit-blasted or spray painted. Two identical AIRs that have the same capabilities are deployed. Assuming that the base positions of the two AIRs are carefully calculated such that each AIR can cover all surfaces of the I-beam, then as shown in Fig. 2a, the paths generated to cover the surfaces of the I-beam would be the same for both AIRs. However, this assumption might not always hold true since the AIRs may have different capabilities and their base positions may not be carefully calculated. Let the surface areas that can be reached by only one of the AIRs be called *specific areas* and the paths created on these areas be called *specific paths*. Similarly, let the surface areas that both AIRs can reach be called *overlapped areas* and the paths created on these surfaces be called *overlapped paths*. For this example, the specific and overlapped paths associated with the two AIRs are shown in Fig. 2b, c. Thus, the problem is to partition and allocate the overlapped area amongst the AIRs such that AIR team's objectives are optimized.

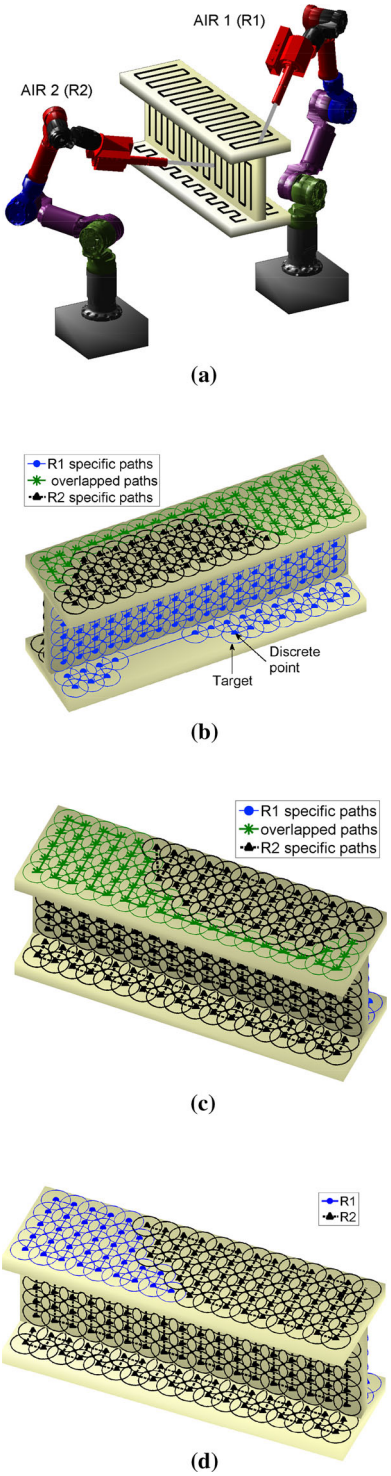


Fig. 2 Overlapped and specific areas as well as the final paths associated with two AIRs which will be used to grit-blast or spray paint an I-beam. **a** Two AIRs grit-blasting or spray painting an I-beam. **b** Overlapped and specific areas (view angle 1). **c** Overlapped and specific areas (view angle 2). **d** Final paths associated with each AIR

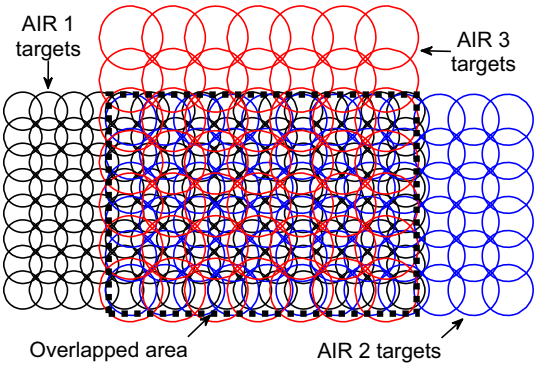


Fig. 3 Three AIRs with different capabilities, each associated with a different set of targets, are used to grit-blast a flat plate

As illustrated in Fig. 2b, a *target* is a circular disk created around a discrete point, such that the discrete point is in the center of the target. The paths can be generated using a suitable CPP algorithm that appropriately links the targets together. To generate targets from the point cloud that is obtained from the sensor/s during the exploration and mapping stage, the method explained in Paul et al. (2013) is used. In brief, the method includes dividing the environment into equally sized, cube-shaped volumetric pixels (voxels), and then associating each voxel with the points that are inside it. A point that is approximately in the center of each voxel is then used as the center point of a sphere with a predefined radius. The plane of best fit for the points within each sphere is then obtained to create a target, and Principal Component Analysis (PCA) is used to find the target's normal vector. Note that this procedure is suitable for field applications where the environment is unstructured. The density and the radius of the targets are chosen to suit the application under consideration. Depending on the application and the type of AIR used, factors such as AIR's speed and the properties of the tool attached to the AIR's end-effector need to be taken into account in determining the targets' size. For example, assume that the paths of three AIRs that have different capabilities are as shown in Fig. 3. The AIRs are used for surface cleaning or maintenance. AIR 3 is expected to be equipped with a tool that can, at each step, cover more of the surface than the other two AIRs, since the targets for this AIR are larger in size.

Let a_{ij}^t be the surface area covered by the target $\boldsymbol{o}_{ij} \in O_i$ where i and j are the AIR index and target index, respectively, and O_i is a set containing all the targets associated with the ith AIR. In order for an AIR to be able to *cover* the target area a_{ij}^t, at least one feasible AIR pose $\boldsymbol{q}_{ij}^f = [\theta_1, \theta_2, \ldots, \theta_{n^k}]$ needs to be found that can reach the target o_{ij} with acceptable end-effector position and orientation, where θ_k is the angular position if the kth joint is revolute or distance if the kth joint is prismatic, and n^k is the total number of joints of the ith AIR. The target $\boldsymbol{o}_{ij}$ is in the specific areas of the ith AIR if no other AIR is able to cover the area a_{ij}^t, otherwise $\boldsymbol{o}_{ij}$ is in the overlapped areas of the ith AIR.

The aim of the approach, besides tackling the problem of partitioning and allocating the overlapped areas, is to ultimately achieve optimal complete coverage of the object's surfaces with the help of a suitable CPP algorithm. The final paths associated with an AIR are to be appropriately created on the specific areas and the portion of the overlapped areas allocated to the AIR (henceforth will be referred to as *allocated areas*). Continuing with the example, it would be expected that the output of the approach will help in modifying the paths shown in Fig. 2b, c, such that for each AIR, the path from the allocated areas is smoothly continued to the specific areas of each AIR as shown in Fig. 2d.

Any approach that is developed needs to take the following requirements into consideration:

1. Prevent overlapped paths: Unlike the condition that is shown in Fig. 4a, the approach is to prevent any section of the overlapped paths remaining overlapped. Overlapped paths can cause longer completion time of the task, higher chance of collision between AIRs and may cause material wastage such as grit in the grit-blasting application.
2. Prevent the path of an AIR from crossing another AIR's path: Two types of crossed paths are shown in Fig. 4b, c. In type 1 shown in Fig. 4b, the paths of the two AIRs cross each other, which may cause a higher chance of collision. In type 2 shown in Fig. 4c, the crossed paths may prevent the AIRs from maneuvering more freely during the task execution since the manipulators of the AIRs will be operating in closer proximity to each other in these regions.
3. Prevent missing sections: In Fig. 4d, some sections of the overlapped areas are not covered by any AIR, that is, *missing sections* are created. The approach should not cause any missing sections.
4. Minimal overall completion time of the task: The approach should minimize the makespan or in other words, the overall completion time of the task. An example of poor overall completion time is shown in Fig. 4e, in which AIR 1 covers a greater portion of the surface.
5. Improve the grouping of the targets: Unlike the condition that is shown in Fig. 4f, the targets in the allocated areas of an AIR should be closely grouped with the targets representing the specific areas of the AIR, i.e. the allocated areas should be as close to their corresponding AIRs as possible. Depending on the application, this requirement can ensure that the motion of an AIR is smoother while transiting from the specific areas to the overlapped areas, the trajectory of the AIR is less constricted by other AIRs, and the likelihood of collision is minimal.
6. Maximal manipulability measure: It is preferred that the approach is capable of optimizing the performance of

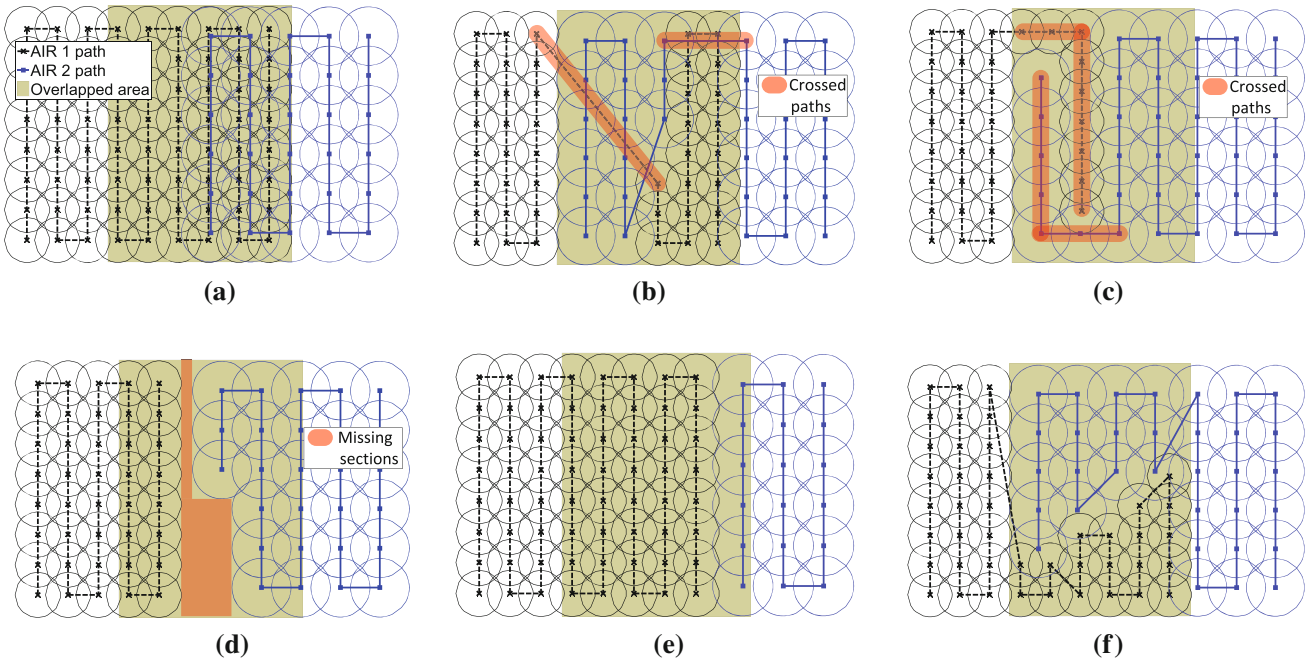


Fig. 4 Examples of unacceptable solutions where in each example at least one requirement is not met. **a** Overlapped paths. **b** Crossed paths (type 1). **c** Crossed paths (type 2). **d** Missing sections. **e** Poor coverage time. **f** Allocated areas are not grouped appropriately with the specific areas

the manipulators attached to the AIRs. For example, assume that multiple solutions satisfy all of the requirements above, then the solution that produces the best manipulator performance should be chosen. A parameter for measuring the performance of a manipulator is the manipulability measure. Maximizing the manipulability measure for a robot manipulator can help the manipulator to make the most use of its degrees of freedom and maneuver more freely in all directions.

7. Minimal torque: It is also preferred that the approach minimizes the torque experienced by each AIR's joints during the task execution.

4 The APA approach

An approach is presented in this paper that simultaneously partitions the overlapped areas and appropriately allocates the partitioned areas to the AIRs in a manner that will optimize the team's objectives. This approach is named Area Partitioning and Allocation (APA).

Note that the style of the notations used are as follows: sets and functions are represented as uppercase letters, scalars are represented as lowercase letters, and vectors or matrices are represented as bold letters. Superscripts are used to help describe the parameter, whereas subscripts are used as indices.

The flowchart in Fig. 5 shows an example of the overall planning process in manufacturing tasks that require AIRs

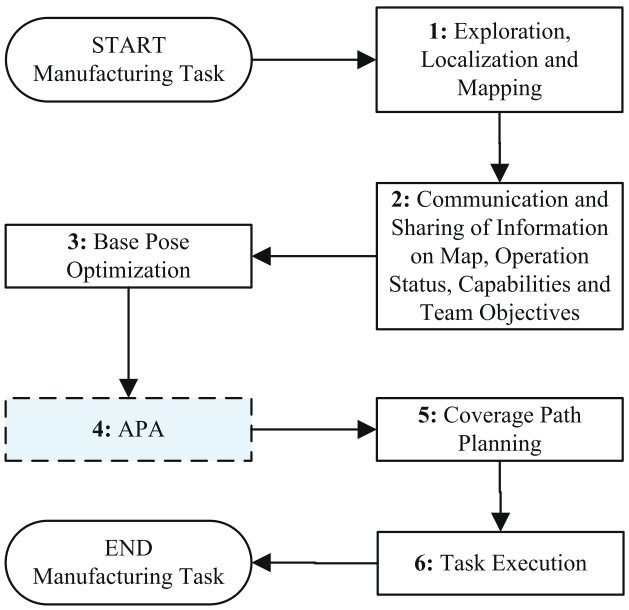


Fig. 5 A flowchart showing the overall planning process and the stage in which APA is carried out

to perform complete coverage in an unstructured environment, such as grit-blasting and spray painting. This paper mainly focuses on component 4 (APA). Other components were explained briefly in the introduction section of the paper. After performing APA, single robot CPP (component 5) such as boustrophedon decomposition (Batsaikhan et al. 2013) or those mentioned in Galceran and Carreras (2013) can

Algorithm 1 Area Partitioning and Allocation (APA)

1: $[O^o, O^s] \leftarrow FindOverlappedAreas(O)$
2: $Z \leftarrow \{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \ldots, \boldsymbol{p}_n^s\}$
3: $Y^p \leftarrow MultiObjOpt\big(Z, ObjFunc(Z, O^o, O^s)\big)$
4: $Y^f \leftarrow SelectFinalSolution(Y^p)$
5: **return** Y^f

be implemented by each AIR to cover the specific and the allocated areas. Motion planning is performed before the execution of the task (component 6), which is outside the scope of this paper.

Algorithm 1 shows the overview of the APA approach. Depending on the size of the surfaces that need to be covered and AIRs' base position and orientation, the portion of the environment that is *reachable* by each AIR can be different, and hence each AIR is associated with a different set of targets, $O_i = \{\boldsymbol{o}_{i1}, \boldsymbol{o}_{i2}, \ldots, \boldsymbol{o}_{in^O}\}$ where n^O is the number of targets associated with the ith AIR. As mentioned previously, in order to label a target $\boldsymbol{o}_{ij}$ as *reachable*, a feasible and collision-free AIR pose $\boldsymbol{q}_{ij}^f$ that can reach the target with appropriate end-effector position and orientation needs to be found, e.g. using the lookup table in Hassan et al. (2015). The first step of the approach is to find the overlapped areas of each AIR, and line 1 of Algorithm 1 depicts this purpose. The function $FindOverlappedAreas$ takes the reachable targets, $O = \{O_1, O_2, \ldots, O_n\}$ associated with the n deployed AIRs and returns the targets, $O^o = \{O_1^o, O_2^o, \ldots, O_n^o\}$ and $O^s = \{O_1^s, O_2^s, \ldots, O_n^s\}$, which represent the overlapped and specific areas of the AIRs, respectively. In order to find the overlapped areas, i.e. to find O^o, distance checking between the targets in O can be performed to determine the targets that overlap.

The APA approach makes use of Voronoi partitioning (Okabe et al. 2008) to partition the overlapped areas into n Voronoi cells, where n is the number of AIRs. Each cell is allocated to an AIR, and the size of the cells is dependent on the location of the seed points, as will be explained in more details in Sect. 5.1. Thus, as shown in line 2 of Algorithm 1, the seed points $\{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \ldots, \boldsymbol{p}_n^s\}$ are considered to be the design variables of the multi-objective optimization problem. The advantage of using Voronoi partitioning is that it reduces the number of design variables. It can also satisfy requirements 1–3 mentioned in Sect. 3, i.e. it can prevent overlapped paths, crossed paths and missing sections. There is, however, a special condition where Voronoi partitioning fails to prevent missing sections when more than two AIRs are deployed. This issue is discussed in Sect. 6.5 and a solution to the problem is presented.

The advantage of using multi-objective optimization (Zhou et al. 2011) is that multiple conflicting objectives can be minimized and a solution that is most suitable for the application under consideration can be chosen from the Pareto front. These advantages are particularly useful for AIRs since other objectives relevant to the manipulator such as torque and manipulability measure can be taken into account. Explanations of how the objectives conflict with each other are provided within the mathematical modeling (Sect. 5.2). Line 3 of Algorithm 1 is to carry out multi-objective optimization using the function $MultiObjOpt$. The Pareto front Y^p is the output of the optimization and is a set of Pareto optimal solutions. Selecting a final solution Y^f from the Pareto front (line 4 of Algorithm 1) is dependent on the priority of the objectives determined based on the application being considered. A selection strategy for selecting Y^f is explained for a particular application in Sect. 6.4.

5 Mathematical Modeling

5.1 Design variables

As aforementioned, the position coordinates of the seed points of the Voronoi cells are considered to be the design variables of the optimization problem. Using Voronoi partitioning the overlapped areas are partitioned into n Voronoi cells, $C^v = \{c_1^v, c_2^v, \ldots, c_n^v\}$ where the cell c_i^v corresponds to the ith AIR and is associated with a seed point position $\boldsymbol{p}_i^s$. Figure 6a illustrates overlapped areas and overlapped paths of two AIRs. The overlapped areas can be partitioned using Voronoi partitioning as shown in Fig. 6b, c. It can be seen that the targets in each Voronoi cell are closest to the seed point of the cell and hence, changing the position of the seed points changes the size and the shape of the cells. Thus, the design variables are

$$Z = \{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \ldots, \boldsymbol{p}_n^s\} \tag{1}$$

as shown in line 2 of Algorithm 1. The goal is to obtain seed point positions and create Voronoi cells that will optimize team's objectives.

5.2 Design objectives

Four objectives are designed to act as the AIR team's objectives. The equations used and the reason for considering such objectives are explained in the following subsections.

5.2.1 Objective 1: Minimal overall completion time

An important objective for applications that require complete surface coverage is minimal completion time or makespan. It is also important that the AIRs can finish with minimal difference in completion times so as to achieve fair workload division amongst the AIRs. As a result, the first objective is to minimize the variance of the completion times of the AIRs:

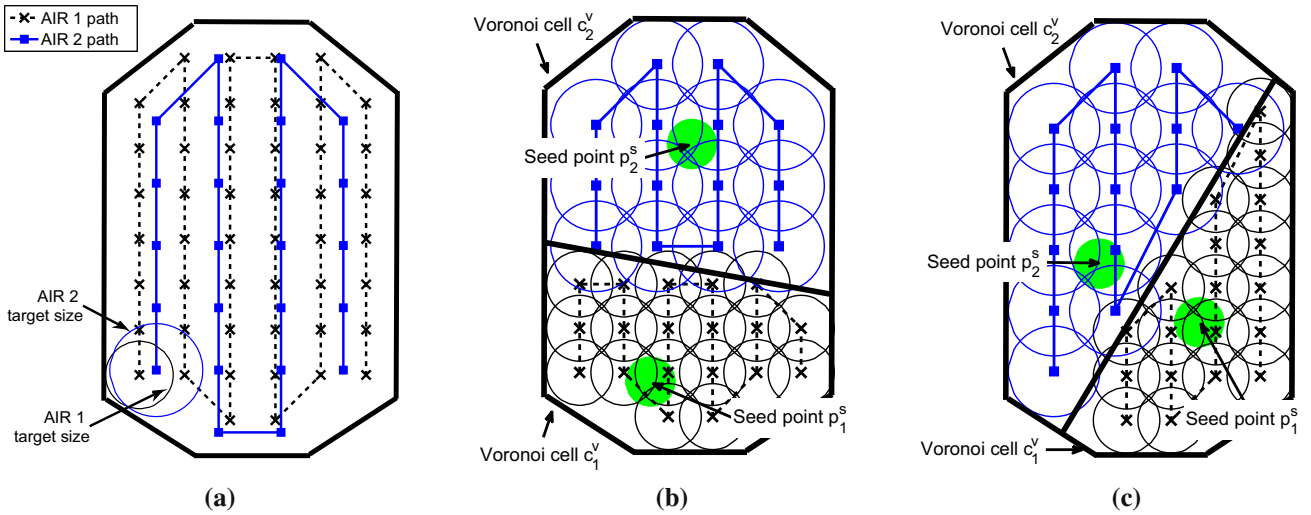


Fig. 6 Two examples of Voronoi partitioning where the overlapped area of two AIRs is partitioned based on the locations of the seed points. **a** Overlapped areas of two AIRs. **b** Voronoi partitioning example 1. **c** Voronoi partitioning example 2

$$\min_Z F_1(Z) = \frac{1}{n}\sum_{i=1}^{n}\left(T_i(Z) - \bar{t}\right)^2 \tag{2}$$

where $\bar{t}$ is the average of the completion times of the n AIRs, and $T_i(Z)$ returns the completion time of the ith AIR. The time $T_i(Z)$ can be expressed as

$$T_i(Z) = \frac{L_i^o(Z) + L_i^s}{v_i} \tag{3}$$

where $L_i^o(Z)$ and L_i^s calculate the lengths of the paths generated on the overlapped and specific areas of the ith AIR, respectively, and v_i is the ith AIR's end-effector speed relative to the surface.

For each AIR, the distance d_i between any two adjacent targets along a path is assumed to be the same and hence,

$$L_i^o(Z) = d_i \cdot \left(N_i^o(Z) - 1\right) \tag{4}$$

and

$$L_i^s = d_i \cdot \left(N_i^s - 1\right) \tag{5}$$

where $N_i^o(Z)$ and N_i^s return the number of targets along the paths of the ith AIR, created on the overlapped and specific areas, respectively. Note that the assumption of constant distance d_i is reasonable, however for certain applications, it may be more accurate to integrate a coverage path planner to obtain an exact length of the path.

Minimal difference in completion times of AIRs will result in optimal makespan due to the fixed coverage area. Minimizing the difference in completion times has the added benefit of achieving fair workload division between the AIRs. This is because even though the makespan can be minimal, the difference in completion times of some AIRs can be large, and it is better to minimize this difference if fair workload division is expected. The makespan and the difference in completion times may not be minimized the same way in the optimization. However, as will be shown in the case studies, if the difference in completion times is relatively small, then the makespan would be near-optimal. As an alternative to Eq. 2, the equation below can be used which purely focuses on minimizing the makespan:

$$\min_Z F_1(Z) = \max\left\{T_1(Z), T_2(Z), \ldots, T_n(Z)\right\}. \tag{6}$$

5.2.2 *Objective 2: Minimal closeness of the allocated areas to the specific areas*

If the areas allocated to the ith AIR are closer to other AIRs than the ith AIR itself, then the motion of the ith AIR can be effected by another AIR or irregular motions of the AIR can be caused to avoid a potential collision. Thus, for each AIR, the closeness of the allocated areas to the specific areas should be minimized. An advantage of having the allocated areas adjoining the specific areas is that the final path can smoothly join the path of the specific areas to the path of the allocated areas, hence avoiding crossed paths. Minimizing the closeness can be done by minimizing the sum of distances between each target $\boldsymbol{o}_{ij}$ (in the set of targets that represented the allocated areas O_i^{al}) and the centroid of the specific areas $\boldsymbol{c}_i^s$ associated with the ith AIR. Thus, the second objective is:

$$\min_Z F_2(Z) = \sum_{i=1}^{n}\sum_{j=1}^{N_i^o(Z)}\left\|\boldsymbol{c}_i^s - \boldsymbol{o}_{ij}\right\|. \tag{7}$$

If the specific areas do not exist for an AIR, then the AIR's base location can be chosen instead of $\boldsymbol{c}_i^s$. Note that targets are created on the surfaces as per the explanation provided previously. For some areas of an object where complex geometries may cause the notion of the normal vector to be unavailable, e.g. sharp edges, target representation is not considered. That is, targets are created on the surfaces only where the normal vector of a target can be computed.

Objective 2 can be in conflict with Objective 1 (minimal makespan). As an example, assume there are multiple AIRs with different capabilities or different specific area size and are deployed to execute a complete coverage task. A faster AIR or an AIR with a smaller specific area may be allocated a larger portion of the overlapped areas in order to achieve minimal makespan. However, this allocation can cause the value of Objective 2 to be large since the sum of distances of the target from the allocated areas to the specific areas of this AIR can be large. Thus, a compromise solution may be necessary.

5.2.3 Objective 3: Minimal AIRs' joints torque

Minimizing the torque experienced by the joints of each AIR during the task execution will, in the long term, help with maintaining the good condition of the manipulator's joints and possibly reduce the energy consumption. There can be certain regions of the overlapped areas that can cause a particular AIR to experience significant torques while covering such regions, and hence these regions are preferred to be covered by another AIR that will not experience large torques.

Let $\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$ be the function that calculates the maximum torque ratio experienced by a joint $k (k = 1, \ldots, n^k)$ of the ith AIR, where $\boldsymbol{q}_{ij}^f$ is a feasible AIR pose that can reach the target $\boldsymbol{o}_{ij} \in O_i^{al}$ with an acceptable end-effector position and orientation, and j is the AIR's pose index as well as the target's index. $\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$ can be expressed as

$$\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f) = \max_k \left| \frac{\mathcal{T}_{ik}(\boldsymbol{q}_{ij}^f)}{\tau_{ik}^c} \right| \tag{8}$$

where $\mathcal{T}_{ik}(\boldsymbol{q}_{ij}^f)$ gives the torque experienced by joint k of the ith AIR at pose $\boldsymbol{q}_{ij}^f$, and τ_{ik}^c is the toque capacity of the kth joint.

For each AIR pose $\boldsymbol{q}_{ij}^f$ corresponding to a target $\boldsymbol{o}_{ij}$, the torque experienced by all joints of the ith AIR can be calculated based on the external forces exerted on the AIR and the weight of the AIR. External forces are mainly the forces at the end-effector. For example, in the grit-blasting application, the reaction force of the blasting stream exiting the nozzle is considered as the external force. Since the motion of the AIR during task execution is slow in such applications, other factors such as angular, centripetal and Coriolis accelerations can be neglected when calculating torque; otherwise, dynamic forces should be taken into account.

The third objective is therefore to minimize the sum of all the maximum torque ratios corresponding to all the AIR poses generated for the targets allocated to each AIR, i.e.

$$\min_Z F_3(Z) = \sum_{i=1}^{n} \sum_{j=1}^{N_i^o(Z)} \mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f). \tag{9}$$

Objective 3 can be in conflict with Objective 1 (minimal makespan). Each AIR will need to be allocated a certain number of targets from the overlapped areas to achieve minimal makespan or difference in completion times. However, to achieve minimal torque it may be better for an AIR to cover certain areas allocated to another AIR since the other AIR may be operating on regions where it is experiencing a large amount of torque. Objective 3 can also be in conflict with Objective 2 since the targets that are close to the specific areas of an AIR may not necessarily cause the AIR to experience less amount of torque, e.g. due to an obstacle that is close to the specific areas and reaching the targets around the obstacle with acceptable AIR poses may cause large torques.

5.2.4 Objective 4: Maximal manipulability measure

Manipulability measure (Yoshikawa 1985) is a measure that can be used to assess the manipulating ability of a robotic system to carry out a task. This measure indicates how far the AIR's manipulator pose is from singularities. This measure is particularly helpful when all joints of the manipulator are revolute or prismatic and not a combination of both (Patel and Sobh 2015). Manipulability measure of an AIR pose $\boldsymbol{q}_{ij}^f$ can be calculated as

$$W(\boldsymbol{q}_{ij}^f) = \sqrt{\det\left(\mathbf{J}(\boldsymbol{q}_{ij}^f)\mathbf{J}^{\mathsf{T}}(\boldsymbol{q}_{ij}^f)\right)} \tag{10}$$

where $\mathbf{J}$ is the Jacobian matrix associated with the pose $\boldsymbol{q}_{ij}^f$.

The fourth objective is to maximize the sum of the manipulability measures, i.e.

$$\max_Z F_4(Z) = \sum_{i=1}^{n} \sum_{j=1}^{N_i^o(Z)} W(\boldsymbol{q}_{ij}^f). \tag{11}$$

Similar to the explanation provided for the previous objective function (Objective 3), Objective 4 can also be in conflict with Objectives 1 and 2. That is, an AIR may be allocated some areas that another AIR can reach with lower manipulability measure; however, this allocation may cause poor makespan since the AIR may already be covering a larger area. Additionally, the targets that are close to the specific

Table 1 Some details of the case studies

Case study	Type of environment	Simulation data or real data	Number of objects	Number of AIRs	Same or different AIRs	Objective functions considered
1	Planar	Simulation	1	3	Different	1 and 2
2	Planar	Real	1	2	Same	Modified
3	3D	Simulation	3	2	Same	All
4	3D	Simulation	3	3	Same	All
5	3D	Real	1	2	Same	All
6	3D	Real	1	2	Same	All

areas of an AIR may not necessarily be reachable with better manipulability measure. Objective 4 can also be in conflict with Objective 3 since an AIR may be able to reach some targets of the overlapped areas with low amount of torque but not necessarily with good manipulability measure, and vice versa.

6 Case studies

Six case studies are presented in this section to validate the approach for different conditions and environments. Table 1 summarizes some details of the case studies. The first case study is to validate the APA approach for three AIRs with different capabilities (speed and tool size). In the second case study, the approach is compared to a pattern-based GA (Genetic Algorithm) which performs partitioning and coverage path planning concurrently. The simulation in this case study is based on a real environment presented in Kapanoglu et al. (2012). Case studies 3 and 4 validate the approach for a complex environment where multiple objects are separated from each other (unconnected), and provide a discussion on the Pareto front. Case studies 5 and 6 use data from a real application and real objects to validate the approach for an object with a complex geometric shape and a concave object, respectively.

The AIRs used in the case studies consist of a nozzle attached to the end-effector of a 6-DOF Schunk industrial manipulator that is mounted on a base. For the case studies with the real-world data (Case Studies 5 and 6), an ASUS Xtion sensor, which is mounted on the nozzle head, was used to generate the point cloud. Computation was carried out on a PC with the following specifications: Windows 7 Enterprise, Intel Core i5-2400 CPU @ 3.10 GHz and 64-bit operating system.

Note that the completion time of an AIR is the time it takes for the AIR to execute the intended task in real-time. It is calculated based on Eq. 3. Thus, the completion time does not include the preliminary computation times of the case study. The computation time associated with the optimization process of APA is included within each case study. A table is added in the next section (Sect. 7) which summarizes the completion times of the AIRs and the computation times for the case studies.

As mentioned in Sect. 4, multi-objective optimization is used. The mathematical model presented in Sect. 5 can be solved using many optimization algorithms. Comparative studies between optimization algorithms and parameters are outside the scope of this paper. An example of multi-objective optimization is Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al. 2000), which is a viable option for the problem under consideration. Thus, the $gamultiobj$ function from Matlab R2013a optimization toolbox, which is based on NSGA-II, is used to solve the mathematical model. In the following subsection, a procedure for calculating the objective functions within the optimization algorithm is provided. For the purpose of improving computation time, tuning of the parameters, "number of generations" and "population size"was investigated with respect to solution quality and convergence. It was found that the population size of 50 is suitable for solution convergence, and the number of generations greater than 100 doesn't improve the solution quality. To demonstrate convergence upon optimal solutions, optimization is repeated 10 times for the relevant case studies and the results are provided. Multithreading is enabled for the optimization to use all four cores of the CPU. All other parameters are set as default.

For a particular AIR pose, the torque experienced by each joint of the AIR needs to be calculated for the objective function expressed in Eq. 9. As mentioned previously, the torque values are due to the external forces and the joints and links weights of the AIR. Specifically, in the following case studies, the torque values account for the reaction force at the nozzle (due to the stream of grit or paint exiting the nozzle), and the weight of the joints (actuators) and the nozzle. The weight of the links is discarded since it is small compared to the actuators' weight. The equations described in Niku (2011) for static force analysis are used to calculate torque. In brief, the torque values due to the forces at a frame H can be calculated as $\mathbf{T}^q(\boldsymbol{q}_{ij}^f) = [\mathbf{J}^H(\boldsymbol{q}_{ij}^f)]^{\mathrm{T}}[\boldsymbol{f}^H]$ where $\mathbf{T}^q(\boldsymbol{q}_{ij}^f)$ gives the torque values of all joints due to the feasible AIR pose $\boldsymbol{q}_{ij}^f$ generated by the ith AIR to reach the jth target,

$\mathbf{J}^H(\boldsymbol{q}_{ij}^f)$ gives the Jacobian matrix relative to the frame H, and $\boldsymbol{f}^H$ contains the forces and moments generated at the frame H. Similarly, torques due to the forces at other frames (frames at the joints and the nozzle) can be calculated. All the torque values can then be added together to give the final torque value of all joints due to the AIR pose $\boldsymbol{q}_{ij}^f$ which is generated to operate on the jth target.

6.1 Procedure for calculating the objective functions

To provide a brief insight into the procedure for calculating the objective functions within the optimization algorithm, Function 1 is constructed. Although NSGA-II is used as an example optimization tool for solving the proposed mathematical model, Function 1 is not limited to genetic algorithms, and can be used with other optimization algorithms provided that small modifications are made if necessary to suit the particular optimization algorithm implemented.

The inputs to the function are the design variables Z (position coordinates of the seed points), the targets $O^o = \{O_1^o, O_2^o, \ldots, O_n^o\}$ that represent the overlapped areas of all AIRs, and the targets $O^s = \{O_1^s, O_2^s, \ldots, O_n^s\}$ that represent the specific areas of all AIRs. The function first loops through the n AIRs (line 2) and all the targets $O_i^o \in O^o$ representing the overlapped areas of the ith AIR (line 3). In each loop it checks (line 4) whether or not a target $\boldsymbol{o}_{ij} \in O_i^o$ is inside the Voronoi cell allocated to the ith AIR by comparing the distance from $\boldsymbol{o}_{ij}$ to the seed point $\boldsymbol{p}_i^s$, with the distance from $\boldsymbol{o}_{ij}$ to every other seed point $\boldsymbol{p}_k^s (k = \{1, 2, \ldots, n\} \setminus i)$. Recall that the targets in a Voronoi cell are closest to the seed point of that particular cell. If the target associated with the ith AIR is inside the corresponding ith Voronoi cell, then the target $\boldsymbol{o}_{ij}$, the maximum torque ratio $\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$ experienced by a joint of the ith AIR at pose $\boldsymbol{q}_{ij}^f$ created to reach $\boldsymbol{o}_{ij}$, and the manipulability measure $W(\boldsymbol{q}_{ij}^f)$ due to the AIR's pose $\boldsymbol{q}_{ij}^f$ are added (lines 5 to 7) to the sets O_i^{al}, $\mathcal{T}_i^{al}$ and W_i^{al}, respectively. Otherwise they will be added (lines 9 to 11) to the sets O_i^{rej}, $\mathcal{T}_i^{rej}$ and W_i^{rej}, respectively. Note that the notation "$\frown$"represents concatenation, and the subscripts al and rej are used to symbolize allocated and rejected targets, respectively. $\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$ and $W(\boldsymbol{q}_{ij}^f)$ are calculated based on Eqs. 8 and 10, respectively. Lines 15 to 18 are designed to deal with a special condition where Voronoi partitioning causes incomplete coverage of the overlapped areas when more than two AIRs are deployed, since the overlapped areas between the AIRs can be different. The function $FixMissingSections$ is designed to fix the missing sections and ensure complete coverage. This function and the method for fixing the missing sections will be explained in detail as part of a case study in Sect. 6.5. After obtaining O^{al}, which contains all the sets of targets allocated to the AIRs, and the corresponding torque values $\mathcal{T}^{al}$ and manipulability measures W^{al}, then objectives F_1 to F_4 (lines 19 to 22) are evaluated using the equations outlined in Sect. 5.2. In line 19, $TimeVariance$ is a function representing Eq. 2. In line 20, $Distance2Centroid$ is a function representing Eq. 7. The Sum used in lines 21 and 22 is to sum all the torque values in $\mathcal{T}^{al}$ and all the manipulability measures in W^{al}, which is the same as evaluating Eqs. 9 and 11, respectively.

Function 1 Calculating Objective Functions

```
1: function ObjFunc(Z, O^o, O^s)
2:   for i = 1 to n do
3:     for j = 1 to n_i^o do
4:       if ||o_ij − p_i^s|| < ||o_ij − p_k^s|| then
5:         O_i^al ← O_i^al ⌢ o_ij
6:         T_i^al ← T_i^al ⌢ T^Rmax(q_ij^f)
7:         W_i^al ← W_i^al ⌢ W(q_ij^f)
8:       else
9:         O_i^rej ← O_i^rej ⌢ o_ij
10:        T_i^rej ← T_i^rej ⌢ T^Rmax(q_ij^f)
11:        W_i^rej ← W_i^rej ⌢ W(q_ij^f)
12:      end if
13:    end for
14:  end for
15:  if n > 2 then
16:    Data ← [O^al, T^al, W^al, O^rej, T^rej, W^rej]
17:    [O^al, T^al, W^al] ← FixMissingSections(Data)
18:  end if
19:  F_1 ← TimeVariance(O^al)
20:  F_2 ← Distance2Centroid(O^s, O^al)
21:  F_3 ← Sum(T^al)
22:  F_4 ← −Sum(W^al)
23:  return [F_1, F_2, F_3, F_4]
24: end function
```

6.2 Case study 1: Three AIRs with different capabilities to grit-blast a flat plate

The purpose of this case study is to demonstrate that the APA approach can produce good results in conditions where AIRs' capabilities are different. Assume that the overlapped and specific areas that were shown in Fig. 3 are associated with three AIRs and are calculated based on the base locations and workspaces of the AIRs. The AIRs have different capabilities, meaning that they have different speed and tool size; therefore, different target size on the surface. The radius of the targets (r^o in meters), the distance between two adjacent targets along a path (d in meters), and the end-effector speed (v in meters per second) associated with each AIR are summarized in Table 2.

Objectives 3 and 4 are discarded in this case study by assigning all targets a value of zero for torque and manipulability measure. A solution from the Pareto front that results in the minimal overall completion time of the task is chosen. The completion time of AIRs 1 to 3 is 13.5, 13.6 and 13.4 s, respectively. It can be seen from Fig. 7 that the allocated areas

Table 2 Properties of the three AIRs

	AIR 1	AIR 2	AIR 3
r^o (m)	0.03	0.04	0.05
d (m)	0.0402	0.0536	0.067
v (m/s)	0.2	0.15	0.1

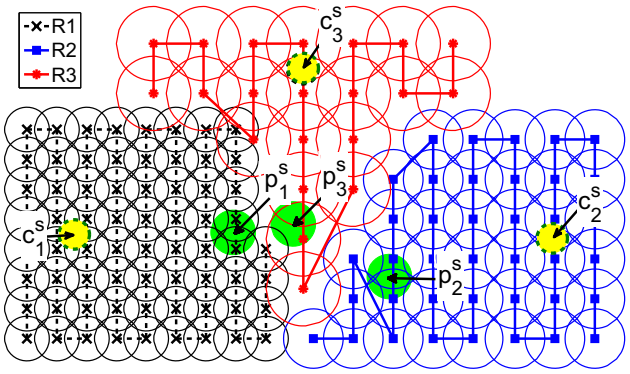


Fig. 7 Final paths associated with the three AIRs based on a solution chosen from the Pareto front

are close to their corresponding AIRs, due to incorporating objective 2 in the optimization model. The total length of the paths of AIRs 1 to 3 is 2.69, 2.04 and 1.34 m, respectively. In the figure, the three filled circles annotated with $\boldsymbol{c}_i^s$ are the centroid of the specific areas associated with the three AIRs where i is the AIR index. Similarly, the three filled circles annotated with $\boldsymbol{p}_i^s$ are the seed points of the Voronoi graph.

To check convergence upon optimal solutions, the optimization was repeated 10 times and for each run, a near-optimal solution was obtained. The above solution is based on one of the 10 optimization runs. From the 10 runs, the average overall completion time of the task is 13.8 s (0.3 s less than the optimal). The computation time for the optimization is <10 s on average.

6.3 Case Study 2: Comparing the proposed APA approach to the pattern-based GA approach

The proposed APA approach focuses on partitioning and allocating the overlapped areas amongst AIRs. It has the flexibility to then allow many of the single robot CPP algorithms to be applied to the allocated areas of each AIR. The APA approach is compared to the pattern-based GA (Genetic Algorithm) approach (Kapanoglu et al. 2012), which performs partitioning and CPP concurrently. In pattern-based GA, eight patterns are designed to "provide disciplined, reasonable rectilinear moves"(Kapanoglu et al. 2012) for each robot. The approach has been validated using flat rectilinear environments, and it has shown to achieve significant improvements over Hierarchical Oriented GA (HOGA).

The 8.4 × 7.2 m environment shown in Fig. 8a has been used in the Pattern-based GA to compare it to the HOGA method. Two scenarios were considered for the comparison where in each scenario different initial start points for the robots are used. The start points are shown in Fig. 8b, c for scenarios 1 and 2, respectively. Kapanoglu et al. (2012) show that using the pattern-based GA, 18% and 14% improvement (over HOGA) is possible for scenarios 1 and 2, respectively. The APA approach is applied to the same scenarios and a further improvement is achieved even though the solution obtained by the pattern-based GA is near-optimal.

For fair comparisons, the objective function is made the same, which is to minimize the overall completion time (makespan). Kapanoglu et al. (2012) consider the completion time of a robot to contain "the linear moving times between

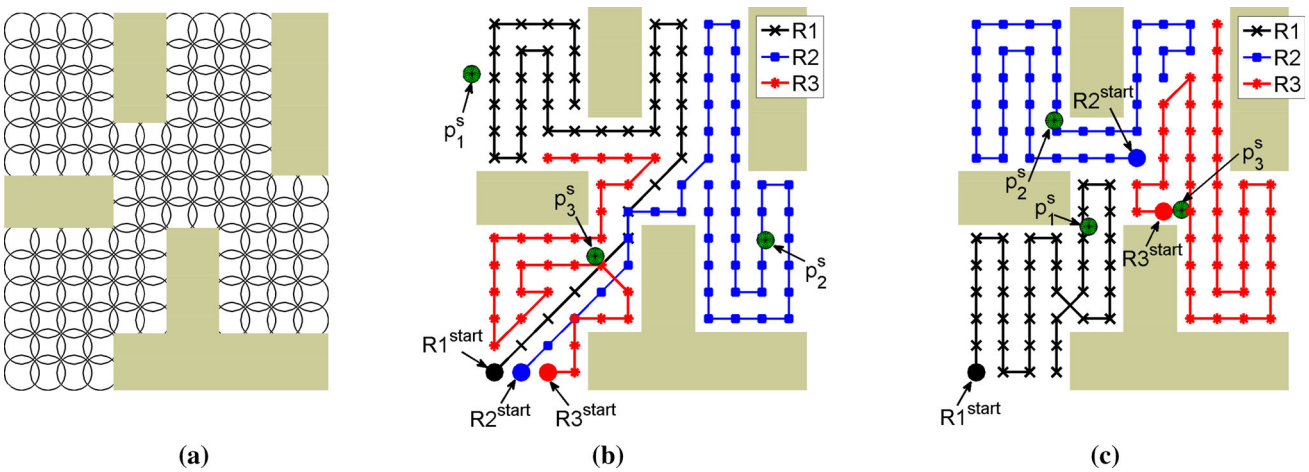


Fig. 8 The test environment used in the pattern-based GA approach, HOGA approach, and the proposed APA approach is shown, and the paths generated using the proposed APA approach is shown for two scenarios where in each scenario different initial start points are considered for the robots. **a** The test environment used for comparisons. **b** Scenario 1. **c** Scenario 2

disk centroids as well as rectilinear turning times", thus the completion time of a robot is calculated as "{[# of straight disk moves]*t_s + [# of right-angle turns]*t_{ra} + [# of rotations (180° turns)] *2*t_{ra}} where t_s and t_{ra} denote the time robot spends moving from one disk to another and the time to make a right angle turn, respectively". In APA, the robot is not restricted to rectilinear moves (right-angle turns and 180° turns), thus the completion time is calculated as: {[path length / length of a straight disk move]*t_s + [sum of all rotations / right angle rotation (90°)]*t_{ra}}. Robots' velocity is made the same, which is 300 mm/s for translation and 20 deg/s for rotation, hence $t_s = 2$ s and $t_{ra} = 4.5$ s.

Using the proposed APA approach the paths shown in Fig. 8b, c are obtained where the overall completion time is 145.9 (∼ 1.5% improvement) and 134.5 (∼ 2% improvement) seconds for scenarios 1 and 2, respectively. Although the improvement may seem small, note that the results presented by Kapanoglu et al. (2012) using the pattern-based GA are already near-optimal. The improvements made by the proposed APA approach is due to the non-rectilinear moves (diagonal moves) of the robots, as can be seen in Fig. 8b, c. The test environment was a rectilinear environment and is ideal for rectilinear moves used in pattern-based GA approach. However, when more complex environments are considered, such as 3D and curved surfaces shown in the following case studies, rectilinear moves may not be efficient, particularly if the cost of the robot turning is high. Hence, the proposed APA approach allows other CPP algorithms to be implemented on the allocated areas of each AIR based on the environment/application. For example, in this case study, an open-end/fixed-start version of the classical traveling salesman problem (TSP) was implemented. Note that in the implementation, robot 1's path is generated first, followed by robot 2 and then 3, and if a robot covers another robot's allocated targets so as to reach its allocated targets, then the already covered targets are removed to avoid repeated coverage as much as possible (e.g. in Fig. 8b).

6.4 Case study 3: Comparing solutions from the Pareto front for two AIRs spray painting three separated objects

The purpose of this case study is to demonstrate that the approach is capable of appropriately partitioning and allocating surface areas of multiple 3D objects that are separated (unconnected) from each other. A comparison study for selecting different solutions from the Pareto front is also presented. As explained in Sect. 4, since multi-objective optimization is used, then a solution from the Pareto front needs to be selected. As an example, assume that the three objects shown in Fig. 9a are to be spray painted by the two deployed AIRs. In selecting a solution from the Pareto front for this application, being able to finish the task in minimal time is the most important priority and hence, objective 1 (minimal overall completion time) is given the top priority, followed by objective 2 (minimal closeness of the allocated areas to the specific areas). Thus, a subset of solutions from the Pareto front is first chosen in terms of objective 1 and then, the set is further reduced in terms of objective 2. From the reduced set of solutions, the final solution can be chosen based on minimizing torque (objective 3) if the joints condition of the AIR is more important than maximizing the manipulability measure (objective 4), or vice versa.

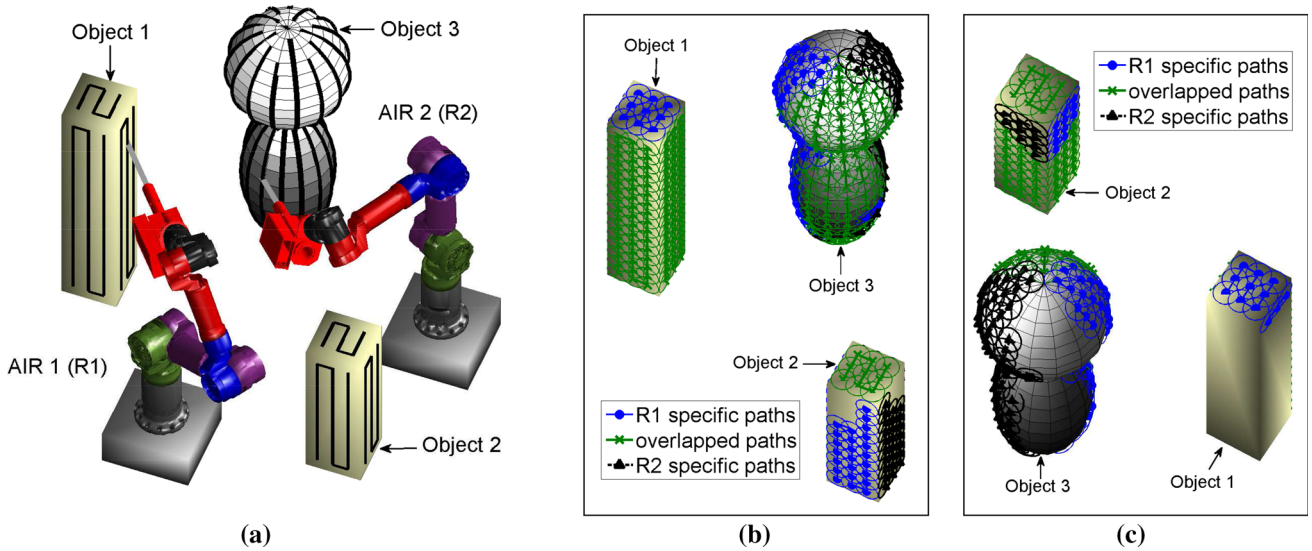


Fig. 9 Overlapped and specific areas as well as the final paths associated with the two AIRs which will be used to spray paint three objects. **a** Two AIRs spray painting three objects which are seperated from each other. **b** Overlapped and specific areas (view angle 1). **c** Overlapped and specific areas (view angle 2)

Table 3 Three solutions from the Pareto front

	Soln. 1	Soln. 2	Soln. 3
Objective 1 (s^2)	0	0	12.1
Objective 2 (m)	142.2	150.6	138.6
Objective 3 (N.m)	120.1	124.5	123.5
Objective 4	−16.4	−16.5	−15.1

Initially, the paths that are shown in Fig. 9a are assumed to be reachable by the two AIRs; however, as explained in Sect. 3, accurate representation of the paths needs to be found by calculating the overlapped and specific areas. The targets and the paths created on the overlapped and specific areas of the two AIRs are shown in Fig. 9b, c. These paths will be modified based on the chosen solution from the Pareto front. The areas that are shown to have no path or targets are unreachable, i.e. they can't be reached with an appropriate end-effector pose of any of the two AIRs.

The two AIRs are assumed to be identical and therefore, both AIRs have the following properties: the radius of the targets (r^o in meters) is 0.04, the distance between two adjacent targets along a path (d in meters) is 0.0563, and the end-effector speed (v in meters per second) is 0.1.

After running the optimization, a small set of solutions from the Pareto front that are best in terms of overall completion time (objective 1) are first chosen. Three of these solutions are shown in Table 3 and Fig. 10, where the spheres annotated with $\boldsymbol{p}_i^s$ are the seed points. Solutions 1 and 2 are optimal in terms of objective 1, i.e. as shown in Table 3,

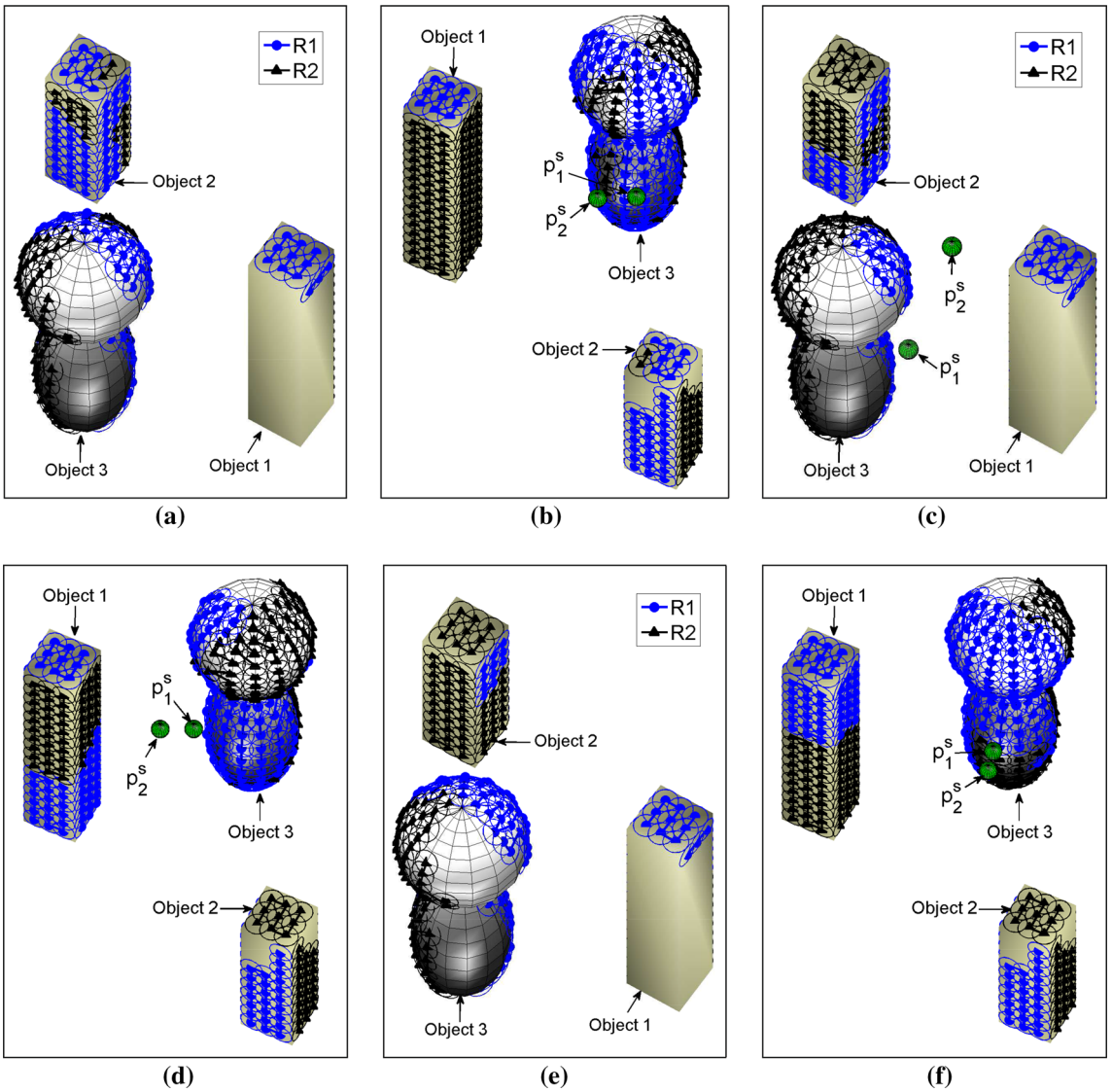


Fig. 10 AIRs' final paths created on the three objects based on three solutions chosen from the Pareto front. **a** Solution 1 (view angle 1). **b** Solution 1 (view angle 2). **c** Solution 2 (view angle 1). **d** Solution 2 (view angle 2). **e** Solution 3 (view angle 1) **f** Solution 3 (view angle 2)

Table 4 Completion time of the AIRs in seconds

	Soln. 1	Soln. 2	Soln. 3
AIR 1 (s)	120	120	117
AIR 2 (s)	120	120	123

the difference in completion times of the AIRs is zero and hence, both AIRs are active during the whole spray painting process. However, choosing solution 1 is better in terms of minimizing torque (objective 3) and minimizing the distance between the allocated areas and the specific areas (objective 2). Maximizing manipulability measure (objective 4) is approximately the same for results 1 and 2. Note that since the optimizer is set to minimize all objectives, then a larger negative value for objective 4 corresponds to a better manipulability measure.

Solution 3 is better than solutions 1 and 2 in terms of objective 2, i.e. the sum of distances from the targets in the allocated areas to the centroids of the specific areas is smaller. Comparing Fig. 10e, f corresponding to solution 3, with Fig. 10a–d corresponding to solutions 1 and 2, it is clear that the targets are better grouped with each other (in Fig. 10e, f) and are more concentrated, meaning that the targets associated with each AIR are better linked to each other to form a path, and will potentially cause a better motion for the AIRs during the task execution.

These comparisons prove that there could be many solutions that result in minimal overall completion time, however by having additional relevant objectives in the optimization model, there is the extra benefit of obtaining better results in terms of one or more of the other objectives and selecting a solution that best suits the application under considerations.

The expected completion time of the two AIRs (in seconds), for each solution is shown in Table 4.

To check convergence upon optimal solutions, the optimization was repeated 10 times, and for each run, an optimal solution was obtained. The average of the optimal solutions selected from the Pareto front of the 10 runs is 0 s^2, 139.9 m, 128.2 N.m, and −15.9, for objectives 1 to 4, respectively. From the 10 optimization runs, the average overall completion time of the task is 120 s (optimal). The computation time for the optimization is <15 s on average.

6.5 Case study 4: Demonstration of a method to fix missing sections when more than two AIRs are deployed

In this case study, the same scenario presented in case study 3 is used; however, an additional AIR that is identical to the other two AIRs is now introduced in the environment as was shown in Fig. 1. An advantage of having more than two AIRs is that more of the objects' surfaces can be covered; however when implementing the APA approach, if an additional procedure is not performed to ensure missing sections are not generated, then incomplete coverage may occur.

Missing sections can be caused only when: (1) more than two AIRs are deployed to carry out the task, and (2) the overlapped areas are not the same for all AIRs. In such a condition, Voronoi partitioning may generate Voronoi cells that may only be partially reachable by their corresponding AIRs and hence, result in missing sections as shown in Fig. 11a, b. This issue can be fixed by first finding and then allocating the missing sections to the AIRs that can reach the sections. In each iteration of the optimization process, Voronoi partitioning is first performed within Function 1 and then the function will check for and fix missing sections by running Function 2.

Function 2 loops through the n AIRs (line 2) and all unallocated/rejected targets, $\boldsymbol{o}_{ij} \in O_i^{rej}$ of each AIR (line 3). In each loop, the area a_{ij}^{rej} that the rejected target $\boldsymbol{o}_{ij}$ covers needs to be checked (line 4) to examine whether or not it has already been covered by another target $\boldsymbol{o}_{i'j'}(\forall \boldsymbol{o}_{i'j'} \in O_{i'}^{al}, i' = \{1, 2, \ldots, n\} \setminus i)$ where $r_{ij}^{\boldsymbol{o}}$ in line 4 is the radius of the target $\boldsymbol{o}_{ij}$. If the condition is met and no other target has covered the area a_{ij}^{rej}, then another check (lines 5 to 7) is performed to examine whether or not a_{ij}^{rej} should actually be given to the target $\boldsymbol{o}_{ij}$ of the ith AIR. In brief, this check is to ensure that the area a_{ij}^{rej} is closest to the ith AIR, or more accurately, closest to the seed point representing the allocated areas of the ith AIR. This check is done by performing the following steps: (1) for each AIR, obtaining (if available) a target $\boldsymbol{o}_{i'j''} \in O_{i'}^{rej}$ that overlaps with $\boldsymbol{o}_{ij}$, i.e. a target that can cover most of the area a_{ij}^{rej} (using the function f_{near} in line 5 which calculates the distance between the targets $\boldsymbol{o}_{i'j''}$ and $\boldsymbol{o}_{ij}$, and places the target $\boldsymbol{o}_{i'j''}$ in the set $Nearest$ if the

Function 2 Fix Missing Sections

```
1: function FixMissingSections(Data)
2:   for i = 1 to n do
3:     for j = 1 to n_i^rej do
4:       if min_{i',j'} ||o_ij − o_i'j'|| > r_ij^o then
5:         Nearest ← f_near(r_ij^o, o_ij, O_i'^rej)
6:         D^min ← min_k ||(o_k ∈ Nearest) − p_k^s||
7:         if D^min > ||o_ij − p_i^s|| then
8:           O_i^al ← O_i^al ⌢ o_ij
9:           T_i^al ← T_i^al ⌢ T^Rmax(q_ij^f)
10:          W_i^al ← W_i^al ⌢ W(q_ij^f)
11:        end if
12:      end if
13:    end for
14:  end for
15:  return [O^al, T^al, W^al]
16: end function
```

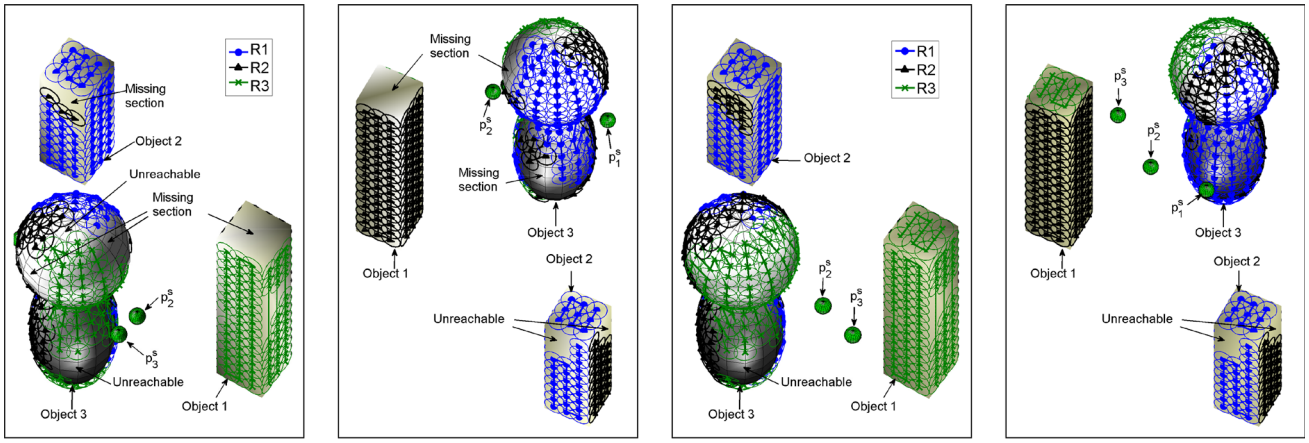


Fig. 11 Two solutions are shown where one of the solutions is not acceptable since missing sections are present, and the other solution is acceptable since all missing sections are found and allocated appropriately. **a** Unacceptable solution (view angle 1). **b** Unacceptable solution (view angle 2). **c** Acceptable solution (view angle 1). **d** Acceptable solution (view angle 2)

distance is less than r^{o}_{ij}, i.e. if $\boldsymbol{o}_{ij}$ overlaps with $\boldsymbol{o}_{i'j''}$), (2) calculating the distances between all targets $\boldsymbol{o}_k \in Nearest$ and their corresponding Voronoi seed point $\boldsymbol{p}^s_k$, and obtaining the minimum distance D^{min} from the calculated distances, and (3) if D^{min} is greater than the distance between the target $\boldsymbol{o}_{ij}$ and its corresponding Voronoi seed point $\boldsymbol{p}^s_i$, then the area a^{rej}_{ij} is allocated to the target $\boldsymbol{o}_{ij}$. Lines 8 to 10 add the target $\boldsymbol{o}_{ij}$, the maximum torque ratio $\mathcal{T}^{Rmax}(\boldsymbol{q}^f_{ij})$, and the manipulability measure $W(\boldsymbol{q}^f_{ij})$, to the sets O^{al}_i, $\mathcal{T}^{al}_i$ and W^{al}_i, respectively. Note that in order to make this function time efficient, the use of bounding volumes or voxels (Xu et al. 2007) and grouping the targets in a hierarchical data structures such as Quadtrees or Octrees (Peters 2013) is needed for reducing the number of distance queries between the targets (e.g. in lines 4 and 5) and acquiring data. Function 2 was detailed for the purpose of clarifications of the implementation only, and may be structured in a different manner. However, regardless of the structure used, the concept remains the same in that, if more than two AIRs are deployed and the overlapped areas are different, missing sections are to be found and appropriately allocated to other AIRs that can cover the missing sections.

Continuing with the example scenario, the additional procedure for fixing missing sections helped with obtaining complete coverage as shown in Fig. 11c, d, which is a solution chosen from the Pareto front. The areas that are shown to have no path or targets are unreachable, i.e. they cannot be reached by any of the AIRs. The completion time of all AIRs is 109 s.

To check convergence upon optimal solutions, the optimization was repeated 10 times, and for each run, an optimal solution was obtained. The above solution is based on the output (Pareto front) of one of the 10 optimization runs. The average of the optimal solutions selected from the Pareto front of the 10 runs is 0 s^2, 201.5 m, 181.8 N.m, and −20.5, for objectives 1 to 4, respectively. From the 10 optimization runs, the average overall completion time of the task is 109 s (optimal). The computation time for the optimization is <2 min on average.

6.6 Case Study 5: Two AIRs used to grit-blast part of a steel bridge

The APA approach is tested by using real data generated from the grit-blasting application where rust and other debris are removed from objects' surfaces. In this application, the grit exits the nozzle with a high pressure and helps with cleaning the surfaces. The deployed AIRs are identical and are equipped with the same end-effector nozzle. The AIRs operate with the end-effector speed set to 0.1 m/s relative to the surface. The radius of the targets associated with all AIRs is set to 0.04 m, and the distance between the centers of two adjacent targets along a path is 0.0563 m. Note that in all figures, due to the size of the objects and the large number of targets associated with each AIR, the paths are not shown to help with a clear visualization of the figures.

For this case study, a cardboard test rig is made to replicate part of a steel bridge as shown in Fig. 12a, where two 6-DOF AIRs are being tested prior to being deployed in the real environment. The environment is explored using sensors attached to the end-effector of the AIR, and then the point cloud data is used to generate the targets shown in Fig. 12b. For each AIR, 4130 targets are used to represent the surfaces. From these targets, the combined number of targets that the two AIRs can actually reach (reachable targets) is 3760 considering that the targets in the overlapped areas are counted once. The positions of the two AIRs relative to the objects are shown in Fig. 12b.

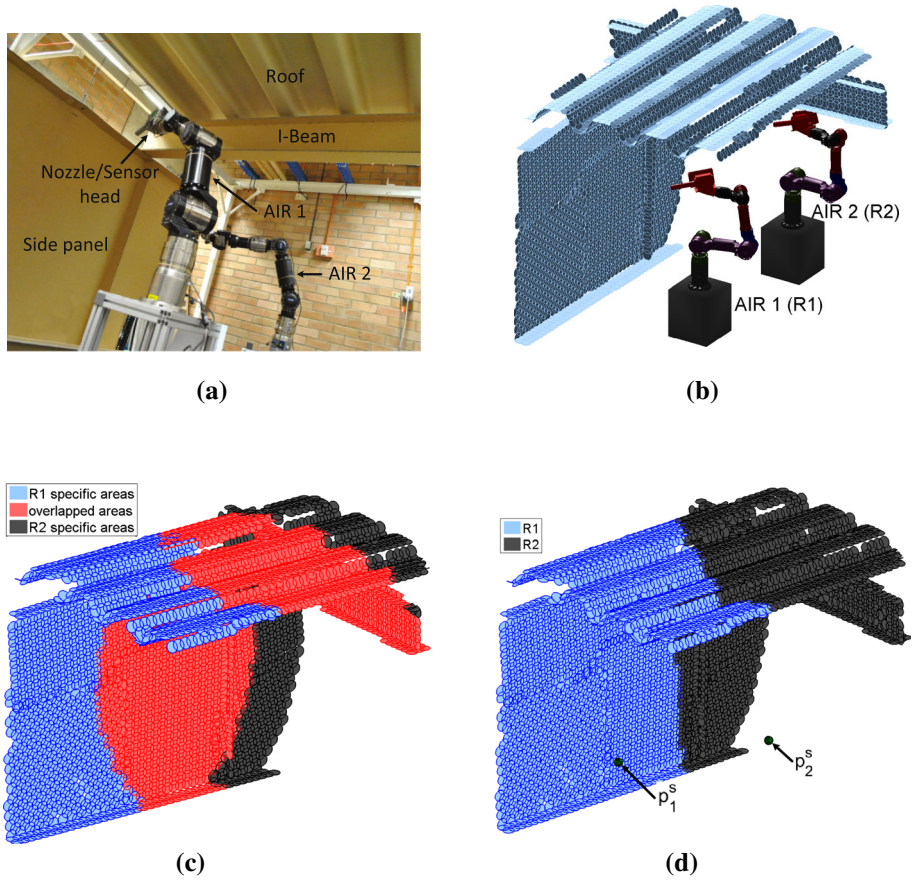


Fig. 12 Overlapped and specific areas as well as the final solution associated with the two AIRs which will be used as a test for a grit-blasting application in a steel bridge environment. **a** Two identical AIRs to be tested in a cardboard test rig replicating part of a steel bridge. **b** Target representation of the environment and the location of the two AIRs. **c** Overlapped and specific areas associated with the two AIRs. **d** Final solution chosen from the Pareto front

Figure 12c shows the specific and overlapped areas associated with the two AIRs. Figure 12d is based on a solution chosen from the Pareto front. This solution is chosen by considering that the overall completion time (objective 1) has the highest priority. The solution is optimal in terms of overall completion time and both AIRs finish simultaneously with a completion time of 969 s. It can be seen that the allocated areas are close to their corresponding specific areas as would be expected by incorporating objective 2.

The optimization was repeated 10 times, and for each run, an optimal solution was obtained. The above solution is based on the output (Pareto front) of one of the 10 optimization runs. The average of the optimal solutions selected from the Pareto front of the 10 runs is 0 s^2, 1870 m, 1225 N.m, and −124, for objectives 1 to 4, respectively. From the 10 optimization runs, the average overall completion time of the task is 969 s (optimal). The computation time for the optimization is <1 min on average.

6.7 Case Study 6: Two AIRs used to grit-blast a boxlike steel structure

The APA approach is tested again using a real concave boxlike steel structure. As shown in Fig. 13a, there are many rusted areas in the structure and two AIRs (same as those used in the previous case study) are expected to be used to remove the rust by grit-blasting all of the internal surfaces of the structure. The targets associated with the two AIRs and the location of the AIRs relative to the structure is shown in Fig. 13b. The specific and overlapped areas, shown in Fig. 13c, d, associated with the two AIRs are made up of a total 6723 targets considering that the targets in the overlapped areas are counted once. It can also be seen from Fig. 13c, d that AIR 2 is able to nearly cover all surfaces, i.e. the targets in the overlapped and specific areas of AIR 2 cover the vast majority of the surface areas. This AIR alone is sufficient to perform the exploration and carry out the grit-blasting. Thus, in this case, introducing a second AIR to the environment is mainly to reduce the overall completion time rather than achieving a greater coverage.

A solution from the Pareto front is shown in Fig. 13e, f. AIR 2 has a larger completion time (1820 s) than AIR 1 (1320 s); however, AIR 1 is allocated all of the overlapped areas to minimize the difference in completion times and the makespan.

The optimization was repeated 10 times, and for each run, an optimal solution was obtained. The above solution is based on the output (Pareto front) of one of the 10 optimization runs. From the 10 optimization runs, the average overall comple-

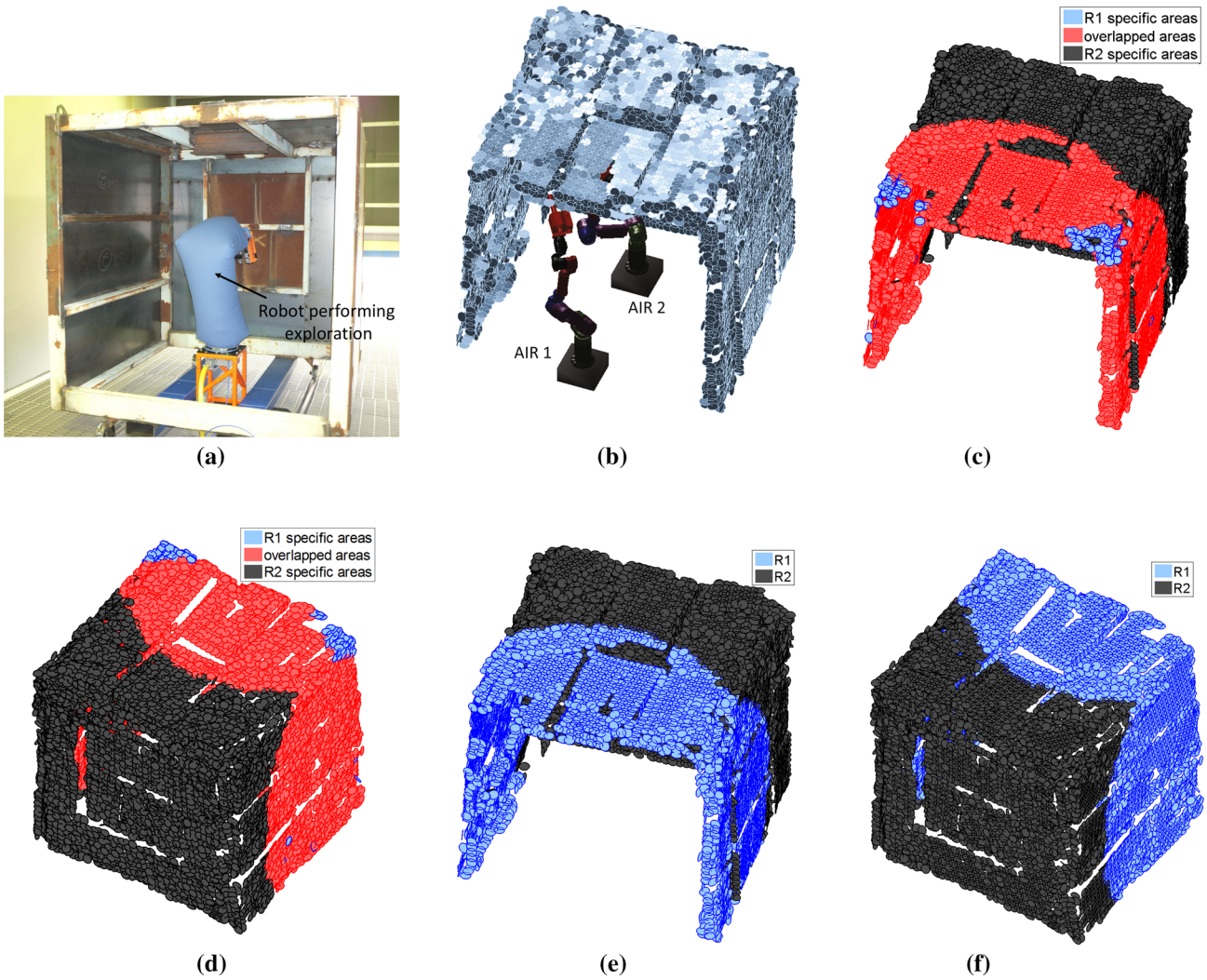


Fig. 13 Overlapped and specific areas as well as the final solution associated with the two AIRs which will be used to grit-blast a convex boxlike steel structure. **a** An AIR performing scanning and exploration of the environment. **b** Target representation and the location of the two AIRs. **c** Overlapped and specific areas (view angle 1). **d** Overlapped and specific areas (view angle 2). **e** Final solution (view angle 1). **f** Final solution (view angle 2)

tion time of the task is 1829 s (9 s less than the optimal). The computation time for the optimization is <2 min on average.

A video is provided (Online Resource) for this case study to recap the steps for obtaining the results. The video also includes the environment exploration phase.

7 Discussion

The case studies demonstrated the effectiveness of the APA approach for various conditions and scenarios. It was shown that the makespan, which is one of the most critical objectives for the applications under consideration, is optimal or near-optimal for all case studies. The benefits of other objectives were also demonstrated in the case studies. Table 5 summarizes the makespan of all case studies. Table 5 also summarizes the computation time of the case studies, i.e. the time it took for the APA approach to find a solution for a particular scenario. The computation times are acceptable for the applications under consideration. However, potential improvements in the computation time can be investigated as

Table 5 Makespan value and computation time of the case studies

Case study	Makespan (s)	Difference from optimal (s/%)	Computation time (s)
1	13.8	0.3/2.2	10
3	120	Optimal	15
4	109	Optimal	120
5	969	Optimal	60
6	1829	9/0.5	120

future work. For example, discretization of the search space can be a way to reduce the computation time by restricting the seed points of the Voronoi graph to be in discretized locations of the environment. The effect of different optimization algorithms on the approach and the mathematical model can also be looked at, and the tuning of the parameters relevant to the chosen optimization algorithm can be studied.

Given a map of the environment, the work presented in this paper primarily focuses on achieving optimal results for the area partitioning and allocation problem. Hence, other stages of AIRs operation such as exploration for mapping were not the main concerns in this paper. The link "http://www.sabreautonomous.com.au" provides relevant resources for better understanding of the AIRs overall operation (e.g. exploration and path planning) for complete coverage task of grit-blasting. The map of each object and the environment can be built using mapping methods by AIR/s if a CAD model of the object is not available, e.g. using the method explained in Paul et al. (2011). Target representation (and their normal) from a point cloud was also explained in Sect. 3. It was assumed that a reasonably accurate map of the environment could be obtained. The accuracy of the generated map that is inputted to the APA approach depends on many factors; e.g. the sensors used, the complexity of the environment, and the exploration and localization algorithm implemented. The accuracy of the map can be improved using methods such as template matching. Inaccuracies and missing data (holes) in the point cloud can also be handled using camera recalibration or depth data filtering (Han et al. 2013). However, these aspects are not investigated in this paper. As future work, it will be interesting to investigate solving the exploration problem and APA concurrently and devise techniques to reduce the inaccuracies.

8 Conclusion and future work

When multiple autonomous robots are deployed in unstructured environments for tasks such as surface cleaning, grit-blasting and spray painting, which require complete surface coverage, it is crucial that the robots are able to collaborate with each other to effectively cover each object that is introduced into the environment, and ultimately achieve optimal operation during the task execution. There are certain areas of the surfaces that more than one robot in the team can reach, which were called overlapped areas in this paper. Prior to generating paths for each robot, the overlapped areas are first to be fairly partitioned and allocated to the robots. In doing so, the capabilities of the robots such as speed, and the robot team's objectives are to be considered. Four objectives were taken into account, and they are as follow: (1) minimal overall completion time of the task, (2) minimal closeness of the allocated areas to the corresponding robot, (3) minimal torque experienced by the robots' joints, and (4) maximal manipulability measure. These objectives were optimized using a multi-objective optimization algorithm. The partitioning of the overlapped areas was carried out using Voronoi partitioning by making the seed points of the Voronoi graph to be the design variables of the multi-objective optimization problem. Several case study were presented to demonstrate the effectiveness of the approach for planar and non-planar objects, multiple objects that are separated from each other, and real objects.

As future work, methods to further improve computation efficiency of the approach can be investigated. It will be also interesting to study the effect of inaccuracies in the point cloud on the APA approach and devise techniques to reduce such inaccuracies. Implementing the approach in a large-scale simulation environment, e.g. using Gazebo, will also be looked at.

Acknowledgements This research is supported by SABRE Autonomous Solutions Pty Ltd and the Centre for Autonomous Systems (CAS) at the University of Technology Sydney (UTS), Australia. Authors thank Prof. Gamini Dissanayake, Assoc. Prof. Shoudong Huang, Dr. Gavin Paul, Dr. Andrew To, Mr. Gregory Peters, and Mr. Teng Zhang for their valuable suggestions and discussions.

References

Batsaikhan, D., Janchiv, A., & Lee, S.-G. (2013). Sensor-based incremental boustrophedon decomposition for coverage path planning of a mobile robot. In S. Lee, H. Cho, K.-J. Yoon, & J. Lee (Eds.), *Intelligent autonomous systems, advances in intelligent systems and computing* (Vol. 193, pp. 621–628). Berlin: Springer.

Carlone, L., Kaouk Ng, M., Du, J., Bona, B., & Indri, M. (2011). Simultaneous localization and mapping using rao-blackwellized particle filters in multi robot systems. *Journal of Intelligent and Robotic Systems*, *63*(2), 283–307.

Danner, T., & Kavraki, L. E. (2000). Randomized planning for short inspection paths. In *IEEE International Conference on Robotics and Automation*, vol 2, (pp. 971–976).

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, & H.-P. Schwefel (Eds.), *Parallel problem solving from nature PPSN VI* (vol. 1917, pp. 849–858)., Lecture Notes in Computer Science, Berlin: Springer.

Englot, B., & Hover, F. S. (2012). Sampling-based coverage path planning for inspection of complex structures. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*. Atibaia, Sao Paulo Brazil. http://hdl.handle.net/1721.1/87729.

Englot, B., & Hover, F. S. (2013). Three-dimensional coverage planning for an underwater inspection robot. *The International Journal of Robotics Research*, *32*(9–10), 1048–1073.

Fazli, P., Davoodi, A., & Mackworth, A. (2013). Multi-robot repeated area coverage. *Autonomous Robots*, *34*(4), 251–276.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258–1276.

Guanghui, L., Yamashita, A., Asama, H., & Tamura, Y. (2012). An efficient improved artificial potential field based regression search method for robot path planning. In *2012 International Conference on Mechatronics and Automation (ICMA)*, (pp. 1227–1232).

Gunady, M. K., Gomaa, W., & Takeuchi, I. (2014). Aggregate reinforcement learning for multi-agent territory division: The hide-and-seek game. *Engineering Applications of Artificial Intelligence*, *34*, 122–136.

Han, J., Shao, L., Xu, D., & Shotton, J. (2013). Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, *43*(5), 1318–1334.

Hassan, M., Liu, D., Huang, S., & Dissanayake, G. (2014). Task oriented area partitioning and allocation for optimal operation of multiple industrial robots in unstructured environments. In *13th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, (pp. 1184–1189).

Hassan, M., Liu, D., Paul, G., & Huang, S. (2015). An approach to base placement for effective collaboration of multiple autonomous industrial robots. In *IEEE International Conference on Robotics and Automation (ICRA*), (pp. 3286–3291).

Hassan, M., Liu, D., & Paul, G. (2016 - in press) Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Janchiv, A., Batsaikhan, D., Kim, B., Lee, W., & Lee, S.-G. (2013). Time-efficient and complete coverage path planning based on flow networks for multi-robots. *International Journal of Control, Automation and Systems*, *11*(2), 369–376.

Kapanoglu, M., Alikalfa, M., Ozkan, M., Yazc, A., & Parlaktuna, O. (2012). A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time. *Journal of Intelligent Manufacturing*, *23*(4), 1035–1045.

Latombe, J.-C. (2012). *Robot motion planning* (Vol. 124). London: Springer.

Maza, I., & Ollero, A. (2007). Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In R. Alami, R. Chatila, & H. Asama (Eds.), *Distributed autonomous robotic systems 6* (pp. 221–230). Japan: Springer.

Niku, S. B. (2011). *Introduction to robotics: Analysis, control, applications* (2nd ed.). Hoboken, NJ: Wiley.

Okabe, A., Boots, B., Sugihara, K., Chiu, S. N., & Kendall, D. G. (2008). *Spatial tessellations: Concepts and applications of Voronoi diagrams* (Vol. 501). London: Wiley.

Patel, S., & Sobh, T. (2015). Manipulator performance measures—a comprehensive literature survey. *Journal of Intelligent and Robotic Systems, 77*(3–4), 547–570.

Paul, G., Webb, S., Liu, D., & Dissanayake, G. (2011). Autonomous robot manipulator-based exploration and mapping system for bridge maintenance. *Robotics and Autonomous Systems*, *59*(78), 543–554.

Paul, G., Kwok, N., & Liu, D. (2013). A novel surface segmentation approach for robotic manipulator-based maintenance operation planning. *Automation in Construction*, *29*, 136–147.

Peters, S. (2013). Quadtree- and octree-based approach for point data selection in 2D or 3D. *Annals of GIS*, *19*(1), 37–44.

Ranjbar-Sahraei, B., Weiss, G., & Nakisaee, A. (2012). A multi-robot coverage approach based on stigmergic communication. In I. Timm & C. Guttmann (Eds.), *Multiagent system technologies* (Vol. 7598, pp. 126–138)., Lecture Notes in Computer Science Berlin: Springer.

Ren, Z., Yuan, J., & Liu, W. (2013). Minimum near-convex shape decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(10), 2546–2552.

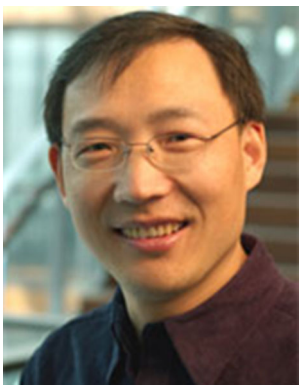
Xu, J., Liu, D., & Fang, G. (2007). An efficient method for collision detection and distance queries in a robotic bridge maintenance system. In T.-J. Tarn, S.-B. Chen, & C. Zhou (Eds.), *Robotic welding, intelligence and automation* (vol. 362, pp. 71–82)., Lecture notes in control and information sciences. Berlin: Springer.

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, *4*(2), 3–9.

Zheng, X., Koenig, S., Kempe, D., & Jain, S. (2010). Multirobot forest coverage for weighted and unweighted terrain. *IEEE Transactions on Robotics*, *26*(6), 1018–1031.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, *1*(1), 32–49.

Mahdi Hassan started his Ph.D. studies at UTS Centre for Autonomous Systems (CAS) in 2013. His research is related to cooperative complete coverage by autonomous industrial robots. His research interests in robotics include complete coverage, path planning, multi-robot collaboration and optimization-based algorithms. He has been involved in projects such as autonomous grit-blasting robots, bio-inspired climbing robots for steel structure inspection, and autonomous underwater robots for inspection of underwater structures.



Dikai Liu received his Ph.D. degree in 1997. His main research interest is robotics including exploration, motion planning, robot teams, and physical human-robot interaction. Professor Liu has been developing novel methods and algorithms that enable robots to operate in unstructured and complex 3D environments autonomously or collaboratively with human users. Example robotic systems he developed and practically deployed include autonomous grit-blasting robots for steel bridge maintenance, assistive robots for human strength augmentation in industrial applications and bio-inspired climbing robots for steel structure inspection.