UNIVERSITY OF TECHNOLOGY SYDNEY

Enabling Methodologies for Optimal Coverage by Multiple Autonomous Industrial Robots

by

Mahdi Hassan

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering and Information Technology
Centre for Autonomous Systems

February 2018

Declaration of Original Authorship

- I, Mahdi Hassan, certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.
- I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This research is supported by an Australian Government Research Training Program Scholarship.

Production Note:
Signature of Student: Signature removed prior to publication.

Date: 23/02/2018

Acknowledgements

Firstly, I would like to thank my principal doctoral supervisor, Prof. Dikai Liu, for providing me the opportunity to carry out this research work and to study in the field of robotics which is of my interest and passion. I appreciate his knowledge and expertise in the field, and I am truly grateful to him for his support and encouragement, and for patiently guiding me through my research.

I thank my co-supervisor, Dr. Gavin Paul, for motivating me with my research through his kind words. I am thankful to him for all his help, for showing interest in my research, and for providing me with valuable advice and reviews to improve the quality of my work.

My gratitude also goes to all project members and in particular Prof. Gamini Dissanayake, Asoc. Prof. Shoudong Huang, Dr. Andrew To, and Mr. Teng Zhang for their valuable discussions and suggestions. A big thanks to all other members of CAS for making the research environment fun and exciting, and for helping out when possible. Many thanks to my friends in the faculty of engineering, particularly Mr. Raphael Falque, Mr. Yuhan Huang, and Mr. Nizar Al-Muhsen for their friendship and support. I would also like to thank UTS, the project partner (SABRE Autonomous Solutions), and CAS for fundings and their interest in my research.

I would like to express my special gratitude to my parents for motivating me to pursue my research. A special thank to my wife Yousra for believing in me, understanding the busy nature of PhD studies and patiently helping with simplifying my life while I undertake my research.

Abstract

Unlike traditional industrial robots which are purpose-built for a particular repetitive application, Autonomous Industrial Robots (AIRs) are adaptable to new operating conditions or environments. An AIR is an industrial robot, with or without a mobile platform, that has the intelligence needed to operate autonomously in a complex and unstructured environment. This intelligence includes aspects such as self-awareness, environmental awareness, and collision avoidance. In this thesis, research is focused on developing methodologies that enable multiple AIRs to perform complete coverage tasks on objects that can have complex geometric shapes while aiming to achieve optimal team objectives.

For the AIRs to achieve optimal complete coverage for tasks such as grit-blasting and spray painting several problems need to be addressed. One problem is to partition and allocate the surface areas that multiple AIRs can reach. Another problem is to find a set of appropriate base placements for each AIR and to determine the visiting sequence of the base placements such that complete coverage is obtained. Uncertainties in base placements, due to sensing and localization errors, need to be accounted for if necessary. Coverage path planning, i.e. generating the AIRs' end-effector path, is another problem that needs to be addressed. Coverage path planning needs to be adaptable with respect to dynamic obstacles and unexpected changes. In solving these problems, it is vital for the AIRs to optimize the team's objectives while accounting for relevant constraints.

This research develops new methodologies to address the above problems, including (1) a Voronoi partitioning based approach for simultaneous area partitioning and allocation utilizing Voronoi partitioning and multi-objective optimization; (2) optimization-based methods for multi-AIR base placements with uncertainties; and (3) a prey-predator behavior-based algorithm for adaptive and efficient real-time coverage path planning, which accounts for stationary or dynamic obstacles and unexpected changes in the coverage area.

Real-world and simulated experiments have been carried out to verify the proposed methodologies. Various comparative studies are presented against existing methods. The results show that the proposed methodologies enable effective and efficient complete coverage by the AIRs.

Contents

List of Figures

List of Tables

Abbreviations

AIMM	**A**utonomous **I**ndustrial **M**obile **M**anipulator
AIR	**A**utonomous **I**ndustrial **R**obot
APA	**A**rea **P**artitioning and **A**llocation
AUV	**A**utonomous **U**nderwater **V**ehicle
CAS	**C**entre for **A**utonomous **S**ystems
CPP	**C**overage **P**ath **P**lanning
DOF	**D**egrees **O**f **F**reedom
FBP	**F**avored **B**ase **P**lacement
GA	**G**enetic **A**lgorithm
MOEA	**M**ulti-**O**bjective **E**volutionary **A**lgorithm
NSGA	**N**ondominated **S**orting **G**enetic **A**lgorithm
OMBP	**O**ptimization of **M**ultiple **B**ase **P**lacements for each AIR
POS	**P**areto **O**ptimal **S**olutions
PPCPP	**P**rey-**P**redator **C**overage **P**ath **P**lanning
RFP	**R**obotic **F**iber **P**lacement
SA	**S**imulated **A**nnealing
SD	**S**tandard **D**eviation
UAV	**U**nmanned **A**erial **V**ehicle
UTS	**U**niversity of **T**echnology **S**ydney

Nomenclature

General Referencing

x	A scalar
$\boldsymbol{x}$	A vector
X	A set
$\boldsymbol{X}$	A matrix
$x^{\cdots}$	Front superscript is part of the notation and is used to help describe the parameter
$x_{\cdots}$	Front subscripts are indices unless mentioned otherwise

General Formatting Style

$\mathrm{F}(\cdots)$	A scalar valued function
$\mathbf{F}(\cdots)$	A vector valued function
$E[\cdots]$	Expected valued function
$[\cdots]^{\mathsf{T}}$	Transpose
$\{\cdots\}$	A set
$\lvert\cdot\rvert$	Absolute value
$\lVert\cdot\rVert$	Vector length
$(\cdot)^n$	A parameter to the power of n
$\mathcal{U}(\cdots)$	Uniform Distribution
$\mathcal{N}(\cdots)$	Normal Distribution

Specific Symbol Usage (Roman Symbols)

A	The surface *areas* representing the overlapped areas of the AIRs
A_i	The surface *areas* from the overlapped areas allocated to the ith AIR
a_{ij}^t	A surface *area* represented by the jth *target*, associated with the ith AIR
B_i	A set of discrete *base* placements for the ith AIR
B_i^{FBP}	A subset of *base* placements from the set B_i, which are called Favored Base Placements (*FPBs*)
$\boldsymbol{b}_{ij}$	The jth discrete *base* placement from the set B_i
C^v	A set containing the *Voronoi cells* of all AIRs
$\boldsymbol{c}_i^s$	The *centroid* of the ith AIR's *specific* areas, i.e. areas that can only be covered by the ith AIR
c_i^v	A *Voronoi cell* representing part of the overlapped areas to be covered by the ith AIR
$\mathrm{D}(\boldsymbol{o}_j)$	A function that calculates the *distance* from the neighbor $\boldsymbol{o}_j$ to the predator
$\mathrm{D}^{max}(\boldsymbol{o}_k)$	A function that calculates the *maximum distance* of the distances from the neighbors of the current prey target to the predator
$\mathrm{D}^{min}(\boldsymbol{o}_k)$	A function that calculates the *minimum distance* of the distances from the neighbors of the current prey target to the predator
d_i	The *distance* between two adjacent targets along a path of the ith AIR
e_i	The maximum anticipated *errors* in the base placement of the ith AIR
$\mathbf{F}(\boldsymbol{P}_i)$	A function that returns the *fitness* values for the ith GA population $\boldsymbol{P}_i$
$\mathrm{F}_j(Z)$	The jth objective *function* which is calculated based on the design variables in Z
$\boldsymbol{F}^H$	The *forces* and moments generated at the frame H
g_{ik}	The kth nonzero *gene* in the ith part of a chromosome
i, j, k, l, m	Used as indices
I^s	A set containing the *indices* of the progress times in T^s
$\mathbf{J}(q_i^f)$	A function that returns the *Jacobian* of the pose $\boldsymbol{q}_i^f$ of the ith AIR
K^{max}	The *maximum* number of observations from a probability distribution which represents uncertainties in a base placement

$\mathrm{L}^c(P^Z)$	A function that calculates the *length* of a path P^Z generated based on the design variables Z and by considering the sequence of, and the distance between, the *covered* targets
$\mathrm{L}_i^o(Z)$	A function that calculates the *length* of a path generated on the *overlapped* areas of the ith AIR based on the design variables in Z
l_i^s	The *length* of a path generated on the *specific* areas of the ith AIR
$\mathrm{N}^N(\boldsymbol{o}_j)$	A function that calculates the *number* of *neighbors* of the jth neighbor of the prey
$\mathrm{N}_i^o(Z)$	A function that calculates the *number* of targets along the paths of the ith AIR that are created on the *overlapped* areas
$\mathrm{N}^f(Z_{ik})$	A function that calculates the *number* of target that can be reached with *feasible* poses of the ith AIR at the kth base placement based on Z_{ik}
$N(\boldsymbol{o}_k)$	A set of *neighbors* of the prey $\boldsymbol{o}_k$
$N^u(\boldsymbol{o}_k)$	A set of *uncovered* and *obstacle-free neighbors* of the prey $\boldsymbol{o}_k$
$N^u(\boldsymbol{o}_j)$	A set of *uncovered neighbors* of the jth neighbor $\boldsymbol{o}_j$ of the prey $\boldsymbol{o}_k$
n	The *number* of AIRs deployed
n_i^b	The *number* of discrete *base* placements in the set B_i
n^c	The *number* of loops where temperature is kept *constant* for the simulated annealing algorithm
n_i^D	The *number* of nonzero genes selected from *dad*'s chromosome for the ith part of a chromosome
n_i^F	The *number* of *favored* base placements (i.e. size of the set B_i^{FBP})
n_i^g	The *number* of *genes* in the ith part of a chromosome (i.e. the length) corresponding to the ith AIR
n^{gen}	The *number* of *generations* for the Genetic Algorithm
n_i^J	The *number* of *joints* of the ith AIR
n^K	The maximum *number* of observations from the distribution that represents uncertainties in a base placement
n^k	The *number* of *steps* associated with a prey's path
n_i^M	The *number* of nonzero genes selected from *mom*'s chromosome for the ith part of a chromosome

n_k^N	The *number* of *neighbors* of the prey at step k
$n^{N_{max}}$	The *maximum* possible *number* of *neighbors* of the prey target
n^O	The *number* of *targets* that represent the surface (if subscript i is added then the targets are associated with the ith AIR)
n^{O^r}	The *number* of *targets* that represent the *reachable* areas (if subscript i is added then the targets are associated with the ith AIR)
n_i^o	The *number* of targets in the *overlapped* areas, which are associated with the ith AIR
n^p	The *population* size for the Genetic Algorithm
n_i^{rej}	The *number* of *rejected* targets of the ith AIR, i.e. the targets in the overlapped areas that are not allocated to the ith AIR
n_i^s	The *number* of targets in the *specific* areas of the ith AIR
n_i^T	The *number* of targets associated with the ith AIR which represent all surfaces irrespective of whether or not the targets can be reached
n^v	The *number* of base placements to be *visited* by all AIRs
n_i^v	The *number* of base placements to be *visited* by the ith AIR
O	A set with a collection of sets where each set contains an AIR's *targets* which represent all surfaces
O_i	A set of *targets* that are associated with the ith AIR and are used to represent all surfaces
O_{ik}	A set of *targets* that represent a surface and are within the workspace boundary of the ith AIR at the kth base placement
O^{al}	A set with a collection of sets where each set contains the *allocated targets* of the ith AIR
O_i^{al}	A set containing the *targets* that are *allocated* to the ith AIR
O_i^c	A set of *targets* that have already been *covered* by the ith AIR
O_k^c	A set of *targets* that have already been *covered* by the prey up-to step k
O^o	A set with a collection of sets where each set contains the *overlapped targets* of an AIR
O_i^o	A set of *targets* that represent the *overlapped* areas of the ith AIR, which more than one AIR can cover

O_k^{ob}	A set which contains all the targets that are predicted to be occupied by *obstacles* at step k
O^r	A set of *targets* that are *reachable* by an AIR with acceptable end-effector pose (if subscript i is added then targets are associated with the ith AIR)
O_{ik}^r	A set of *targets* that represent a surface and are *reachable* from the kth base placement of the ith AIR
O^{rej}	A set with a collection of sets where each set contains the *rejected targets* of the ith AIR
O_i^{rej}	A set of *targets* in the overlapped areas that are not allocated (*rejected*) to the ith AIR
O^s	A set with a collection of sets where each set contains the *targets* of an AIR that represent the *specific* areas
O_i^s	A set of *targets* that represent the *specific* areas of the ith AIR, which only the ith AIR can cover
O_i^u	A set of *targets* that are assigned to the ith AIR but have not been covered (*uncovered*)
O_k^u	A set of *targets* that are not yet covered (*uncovered*) by the prey at step k
$\boldsymbol{o}$	A *target* representing part of a surface
$\boldsymbol{o}_k$	The prey *target* at step k (the prey is defined as the coverage spot of the end-effector tool)
$\boldsymbol{o}_{ij}$	The jth *target* associated with the ith AIR
$\boldsymbol{o}_{ijk}$	The kth *target* that is within the workspace boundary of the ith AIR, and that might be reachable, at the jth base placement
$\boldsymbol{o}_i$	The ith *neighbor* of the prey $\boldsymbol{o}_k$ (from the set $N(\boldsymbol{o}_k)$)
$\boldsymbol{o}_j$	The jth uncovered and obstacle-free *neighbor* of the prey $\boldsymbol{o}_k$ (from the set $N^u(\boldsymbol{o}_k)$)
$\boldsymbol{o}_{j_k^*}$	The *neighbor* of the prey with maximal reward at step k
$\boldsymbol{o}_k^p$	The *preceding target* that was covered by the prey at $(k-1)$th step
$\boldsymbol{o}^s$	The *start target* of the prey
$\boldsymbol{P}_i$	The ith *population* for the Genetic Algorithm
P^Z	A *path* generated based on the values of the design variables Z

$\mathcal{P}$	A chromosome (offspring) within a GA population
$\boldsymbol{p}_i^s$	The *seed point* of a Voronoi cell, which is associated with the ith AIR
$\boldsymbol{q}_i$	A pose of the ith AIR, which is defined by the joints angles of the AIR
$\boldsymbol{q}_{ij}^f$	A *feasible* pose of the ith AIR that reaches the jth target with correct end-effector position and orientation, and without collision
$\mathrm{R}(\boldsymbol{o}_j)$	The total *reward* function associated with the target $\boldsymbol{o}_j$
$\mathrm{R}^s(\boldsymbol{o}_j)$	The *smoothness reward* function associated with the target $\boldsymbol{o}_j$
$\mathrm{R}^b(\boldsymbol{o}_j)$	The *boundary reward* function associated with the target $\boldsymbol{o}_j$
$\mathrm{R}^d(\boldsymbol{o}_j)$	The *distance reward* function associated with the target $\boldsymbol{o}_j$
r	The *radius* of a sphere within which targets are considered to be neighbors of a target/prey
r^o	The *radius* of a *target*
r_{ij}^o	The *radius* of the jth *target* of the ith AIR
$\mathrm{T}_i(Z)$	A function that calculates the overall completion *time* of the ith AIR based on the design variables in Z
$\mathcal{T}_{ik}(\boldsymbol{q}_i^f)$	A function that calculates the *torque* experienced by the kth joint of the ith AIR at pose $\boldsymbol{q}_i^f$
$\mathbf{T}^q(\boldsymbol{q}_{ij}^f)$	A function that calculates the *torque* values of all joints due to the forces at a frame and the AIR pose $\boldsymbol{q}_{ij}^f$
$\mathcal{T}^{Rmax}(\boldsymbol{q}_i^f)$	A function that calculates the *maximum torque ratio* due to one of the ith AIR's joints and the AIR pose $\boldsymbol{q}_i^f$
$\mathcal{T}_i^{al}$	A set containing the maximum *torque* ratios corresponding to the *allocated* targets of the ith AIR
$\mathcal{T}_i^{rej}$	A set containing the maximum *torque* ratios corresponding to the *rejected* targets of the ith AIR
T^s	A set containing the progress *times* of the n AIRs *sorted* from the lowest time to the highest
t	The current execution *time* of the coverage task
$\bar{t}$	The *average* of the completion *times* of the n AIRs
t_i	The current progress *time* of the ith AIR
t^c	The overall *completion time* of the task (makespan)

t_i^s	The *time* associated with the ith AIR *setting-up* and moving to the next base placement
t^{max}	The *maximum time* allocated to the coverage task
v_i	The end-effector speed of the ith AIR
v_i^d	The *difference* between the maximum and the minimum end-effector speed of the ith AIR
v_i^{max}	The *maximum* end-effector speed of the ith AIR
v_i^{min}	The *minimum* end-effector speed of the ith AIR
$\mathrm{W}(\boldsymbol{q}_i^f)$	A function that calculates the manipulability measure of the pose $\boldsymbol{q}_i^f$
W_i^{al}	A set containing the manipulability measure associated with the *allocated* targets of the ith AIR
W_i^{rej}	A set containing the manipulability measure associated with the *rejected* targets of the ith AIR
Y^p	The output of the multi-objective optimization which is a set of solutions on the *Pareto* front
$\boldsymbol{y}^f$	The *final* solution chosen from the Pareto front (i.e. from Y^p)
Z	A set containing the design variables
Z_{ik}	The kth design variable associated with the ith AIR

Specific Symbol Usage (Greek Symbols)

α_i	The cooling ratio for the simulated annealing algorithm, corresponding to the ith objective function
$\boldsymbol{\beta}_i$	A favored *base* placement from the set B_i^{FBP}, associated with the ith AIR
$\boldsymbol{\beta}_i^{AIR}(t)$	The *base* placement of the ith *AIR* at time t
δ	The minimum distance threshold between the base placements of any two AIRs
δ_{ik}^s	A *small* negative or positive integer to be added to the gene g_{ik}
θ_j	The *angle* of the jth joint of an AIR pose
Ξ^z	A set with each element in the set representing the uncertainties associated with an AIR's base placements expressed as a random vector with multivariate normal distribution

$\boldsymbol{\xi}_{ij}$	An observation from a probability distribution which represents uncertainties in the jth base placement of the ith AIR
$\boldsymbol{\xi}^k$	The kth observation from a probability distribution which represents uncertainties in a base placement
$\boldsymbol{\Sigma}$	The covariance matrix associated with a multivariate normal distribution
σ^2	The variance
τ_i	The initial *temperature* for the simulated annealing algorithm, corresponding to the ith objective function
τ^c_{ik}	The torque *capacity* of the kth joint of the ith AIR
$\boldsymbol{\psi}$	The predator location
ω_{ikj}	A weighting factor (from 0 to 1) applied to the end-effector speed of the ith AIR based on the area in which the target o_{ikj} is located
ω^s	A weighting factor for the *smoothness* reward function
ω^b	A weighting factor for the *boundary* reward function

Glossary of Terms

AIR path	The path that an AIR follows by adjusting its joints angles and base position/orientation.
AIR pose	A pose of an AIR defined by its joints angles and base position/orientation.
AIR team's objectives	A set of objectives, formulated as objective functions, that the AIR team aim to optimize. Examples include achieving minimal completion time and maximal coverage.
Allocated areas	Part of the surface areas of interest allocated to an AIR for coverage.
Autonomous Industrial Robot (AIR)	An industrial robot, with or without a mobile platform, that has the intelligence needed to operate autonomously in a complex and unstructured environment. This intelligence includes self-awareness, environmental awareness and collision avoidance.
Base placement	A base location and orientation for an AIR from which it will operate on a surface or part of a surface.
Boundary reward	The reward associated with the prey covering the targets representing the boundary (boundary targets).
Boundary targets	The targets that represent the boundary of the surface as well as the targets that are on the boundary of the uncovered regions, i.e. the uncovered targets closest to the already covered region of the surface.
Complete coverage	The task of covering (operating on) all areas of a surface.

Complete coverage path	A path on a surface of interest that when covered (followed from start to end) by an end-effector tool of an AIR it will result in complete coverage of the surface.
Complex object	A 3D object with complex geometric shape.
Coverage area	The area to be covered (operated on) by the AIRs' end-effector tool, and excludes the area occupied by obstacles.
Covered targets	The targets on the surface that have been covered (operated on) by one or more AIRs.
Deadlock	The situation where the prey arrives at a target where all neighbors are already covered. In this case, the prey needs to repeat coverage of a certain number of targets in order to reach an uncovered target. PPCPP resumes when the prey reaches an uncovered target.
Dynamic environment	An environment where changes can occur, e.g. stationary or dynamic obstacles may become present. Changes in the environment can be unexpected, i.e. prior to real-time implementation it may not be possible to predict the changes.
End-effector	A point, an area, or a tool at the end of an AIR's arm that interacts with the environment, e.g. the blasting spot in the grit-blasting application or the spray spot in the spray painting application.
End-effector pose	The position and orientation of the end-effector relative to a reference frame.
Environment	A space consisting of AIRs, objects to operate on which can be complex or planar, and dynamic or stationary obstacles.
Exploration	The process in which AIRs navigate and explore an unknown (or partially unknown) environment to obtain information about it and build a map.
Favored Base Placement (FBP)	A base placements for an AIR that results in reasonably high coverage of a surface and that is an acceptable distance away from obstacles.

Feasible AIR pose	An AIR pose that can reach a target with appropriate end-effector orientation and position, and without any collision.
Localization	The process of determining the location and/or orientation of an AIR with respect to a reference point or frame.
Makespan	The overall completion time of a task.
Manipulator	In this thesis, a manipulator is an industrial robotic arm which forms part of an AIR.
Manipulability measure	A measure for a manipulator pose which indicates how far the manipulator is from singularities.
Mapping	The process of constructing a map of the environment (including the objects) in which the AIR operates.
Missed-coverage	The condition where part of a surface is not covered by any AIR.
Missing sections	The sections of the surface that are missed due to a special condition where more than two AIRs are deployed.
Neighbor	A neighboring target of the prey (or another target) which belongs to the neighboring set.
Obstacle	A stationary or a dynamic object that an AIR can collide with due to the object being inside the AIR's workspace for a period of time.
Overlapped areas	The surface areas that more than one AIR can reach with feasible AIR poses as a result of AIRs' workspace overlapping.
Pareto front	A set of Pareto optimal solutions, which is the output of a multi-objective optimization algorithm. All Pareto optimal solutions are considered to be equal in terms of optimality.
Planar environment	An environment where the surface or the object to be operated on can be approximated to be flat.
Platform	A mobile or stationary platform on which the AIR's manipulator is fixed.
Prey	The prey is the coverage spot with a size equivalent to the coverage size of an AIR's end-effector tool.

Predation avoidance reward	The reward associated with the prey maximizing its distance from the predator at each step.
Predator	A point represented as a virtual predator that a prey considers avoiding by maximizing its distance from it.
Reachable target	A target that can be reached by a feasible AIR pose.
Smoothness reward	The reward associated with the prey continuing motion in a straight direction.
Specific areas	The surface areas that can be reached, with feasible AIR poses, by one of the AIRs only.
Surface normal	A 3D vector perpendicular to the surface.
Target	A circular disk that represents part of a surface; and is defined using the location of the disk's centroid, the surface normal, and the radius of the disk.
Target normal	A 3D vector perpendicular to the target.
Task execution	The process of executing the planned task (e.g. grit-blasting or spray painting) by the AIRs after all necessary off-line computations or preparations are completed.
Total reward	The total reward associated with the prey moving to one of the neighbors.
Uncovered targets	The targets that are not covered by any AIR.
Unexpected obstacles	The stationary or dynamic obstacles that are initially unknown to the AIR and are detected in real-time during the coverage task.
Unstructured environment	A complex and uncontrolled real-world environment which is similar to human-like environments and is subject to regular changes and inherent uncertainties.
Voronoi cell	A cell that represents part of a surface and is allocated to an AIR. The cell is created using Voronoi partitioning method where an area is divided into n cells based on the location of n seed points.

Chapter 1

Introduction

The fast advancement of robotic technologies is significantly improving productivity and cost-efficiency of many applications as well as employees' health and safety, e.g. in the automotive industry [1] and steel bridge maintenance [2, 3]. As the shift from mass production to custom production is taking place [4], equipping industrial robots with intelligence and enabling them to operate autonomously is beneficial when operating in complex and dynamic environments. For example, an Autonomous Industrial Robot (AIR) that can spray paint objects for custom production, without assistance from human operators, can be a great advantage. An AIR is an industrial robot, with or without a mobile platform, that has the intelligence needed to operate autonomously in a complex and unstructured environment. This intelligence includes self-awareness, environmental awareness, and collision avoidance. If the AIR is attached to a mobile platform, then its definition is the same as Autonomous Industrial Mobile Manipulator (AIMM) [4].

Utilizing multiple AIRs can provide greater capacity and flexibility. As an example, consider the two mobile AIRs shown in Fig. 1.1 deployed for a complete coverage task, such as grit-blasting, on the three objects. For such applications, utilizing multiple AIRs can help minimize the overall completion time of the task and potentially maximize coverage of the target objects.

The grit-blasting application shown in Fig. 1.1 is similar to many applications that require surface preparation, e.g. abrasive blasting, high-pressure cleaning, surface coating, and

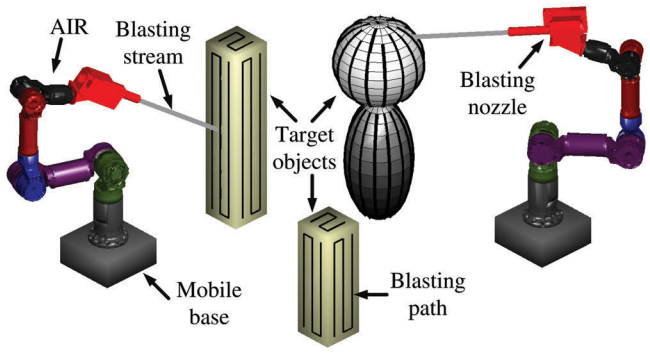


Fig. 1.1: Two mobile AIRs performing grit-blasting on three objects.

spray painting. Complete coverage [5] is an integral part of such applications, i.e. all surface areas of interest need to be operated on to achieve complete coverage. In Fig. 1.1, the stream of grit is directed by the nozzle attached to the end-effector of an AIR, which aims at and follows the paths on the surface of the objects. If both AIRs collectively cover all paths, then complete coverage of the objects is achieved.

A challenge in the deployment of multiple AIRs is to develop methodologies that enable the AIRs to achieve optimal complete coverage. The AIRs need to decide on their base placements, partition and allocate the surface areas amongst themselves, and perform real-time coverage path planning while taking into account changes that occur in the environment. The AIRs need to collectively account for the team's objectives, e.g. achieving minimal completion time and maximal coverage; and team's constraints, e.g. operating without any collisions.

1.1 Motivation

Enabling multiple AIRs to perform a complete coverage task on an object with complex geometric shape and/or in dynamic environments is a challenging problem. However, it is a problem that when solved, can benefit many applications. The need to operate on large objects further complicates the problem. As autonomous robots start to be practically deployed, addressing this complete coverage problem becomes increasingly important.

By utilizing multiple AIRs, the goal can be to improve the productivity and cost-efficiency of the intended task, reduce human exposure to potential health hazards, increase the

capacity of the task, etc. As an example, the steel bridge maintenance task can be considered [2, 3, 6]. As part of this task, grit-blasting is performed on the surfaces to remove old paint, rust, and other debris. Then, spray painting is carried out to protect the surfaces and preserve the integrity of the structure. Both grit-blasting and spray painting are examples of complete coverage tasks. In certain environments and conditions, these tasks can expose the workers to serious health risks and hazards, e.g. grit-blasting often generates dust that contains hazardous substances such as lead and asbestos [3]. With the enormous and ever growing number of steel bridges in the world and the cost associated with maintaining these bridges, it is self-evident how grit-blasting or spray painting AIRs can be beneficial in reducing maintenance cost, increasing productivity, and reducing human exposure to unpleasant and risky environments. In Australia alone, there are over 30,000 bridges [2]. However, bridge maintenance is only one of the many applications that can benefit from the deployment of AIRs.

Traditional industrial robots used for mass production are preprogrammed and purpose-built for a particular repetitive application. Unlike traditional industrial robots, AIRs have the intelligence to carry out the entire task autonomously. Hence, due to their autonomous operation, AIRs have greater flexibility and capacity to perform various tasks, e.g. they can be used for one-off or custom manufacturing and can be deployed to operate in a wider range of applications.

For optimal complete coverage by the AIRs, appropriate methodologies need to be developed. These methodologies are preferred to be modular. That is, it should be possible to use any combination of the developed methodologies so as to adapt to the conditions of the application under consideration. For example, in some applications determining the base locations of the AIRs to cover objects' surfaces is necessary, whereas in other applications the base of the AIRs may be fixed and base placement optimization may not be needed for achieving complete coverage. Additionally, the methodologies need to be constructed by taking account aspects such as uncertainties, dynamic obstacles, unexpected changes in the coverage area, optimization of AIR team's objectives, and satisfying team's constraints. Understanding the challenges of developing such methodologies for multi-AIR complete coverage was the motivation behind the work presented in this thesis.

1.2 Scope

Achieving optimal complete coverage by multiple AIRs is a new research problem that is challenging due to aspects such as the size of the object, the complex nature of the objects' shape, and the presence of dynamic obstacles. The work in this thesis aims to develop methodologies for achieving optimal complete coverage by multiple AIRs while accounting for team's objectives and constraints.

The scope of the work in this thesis is explained with the aid of a flowchart. The flowchart is shown in Fig. 1.2 and gives an abstract illustration of the overall process considered for completing a typical complete coverage task by multiple AIRs. Note that depending on the application under consideration, not all modules may be relevant. The work in this thesis is mainly related to Modules 3 and 4.

The first module shown in the flowchart is exploration for mapping and localization [7–9]. If the AIRs have prior knowledge of the environment, then they start with localizing themselves. However, the robots may not have a full knowledge of the environment; hence full or partial exploration may be needed. As a result, each AIR constructs a partial map of the environment based on its capabilities and location. Then, the AIRs will communicate and share the information they have collected (Module 2 of the flowchart) so as to create a complete map of the environment (including the objects). The AIRs are also expected

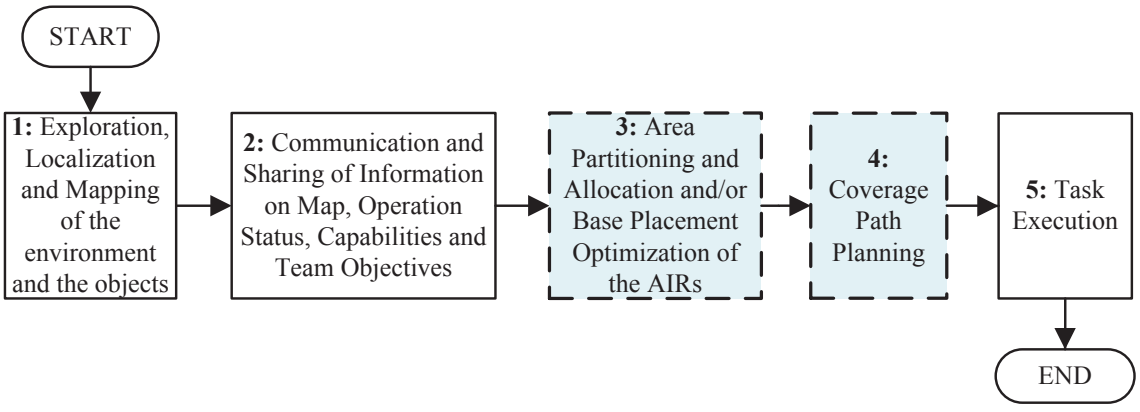


Fig. 1.2: An abstract illustration of a process for conducting a complete coverage task using multiple AIRs.

to have shared information on their capacity, initial position, speed, properties of the end-effector tool, etc. After the first two modules, all AIRs are assumed to have the same knowledge/information on the environment and the objects to be covered.

The relevant information is then passed on to Module 3 to perform simultaneous area partitioning and allocation and/or base placements optimization. Area partitioning and allocation may be necessary since in some circumstances the workspace of multiple AIRs may overlap and as a result, more than one AIR can reach certain areas of objects' surfaces. The base placement optimization problem is to determine an optimal number of base locations on which the AIRs will be placed. For large objects, this problem also includes determining the optimal locations and sequence of the base placements for the AIRs. Next, as per Module 4, each AIR is required to perform coverage path planning [5] on the allocated surface areas at each of its base placements. Coverage path planning is the task of generating a path that covers the surfaces of interest. In this thesis, the coverage path is the AIR's end-effector tool path that the AIR needs to follow so as to cover the allocated surfaces. The tasks in Modules 3 and 4 are carried out while considering AIR team's objectives and constraints. Additionally, these modules collectively take into account aspects such as uncertainties in base placements and unexpected changes in the environment.

Module 5 is related to the task execution. In this thesis, task execution is the process of executing the planned task (e.g. grit-blasting) by the AIRs after all necessary off-line computations or preparations are completed. It may involve collision-free motion/trajectory planning for the AIRs to follow the coverage path generated in Module 4.

The overall process is made modular since not all modules are relevant to all applications. For example, area partitioning and allocation (in Module 3) may not be needed for some applications if the distance between the AIRs can always be kept large enough such that the workspaces of the AIRs never overlap. Another example would be to skip base placement optimization (in Module 3) if the base of the AIRs is fixed, and the object is small and positioned such that the whole object can be covered by the AIRs. The modular nature also provides the flexibility to integrate the modules of interest. For example, if the environment can unexpectedly change during the task execution, then coverage path planning (Module

4) can be combined with task execution (Module 5) to provide the adaptability needed for the AIRs to respond to the changes or the dynamic obstacles present in the environment.

The work presented in this thesis studies the problems in Modules 3 and 4, which are highlighted in the flowchart. Other work, such as exploration, localization and mapping [7–9], and robot or manipulator motion/trajectory planning [10–17] have been widely studied by researchers. Hence, it is assumed that appropriate algorithms can be selected for Modules 1, 2 and 5 that best suit the application under consideration.

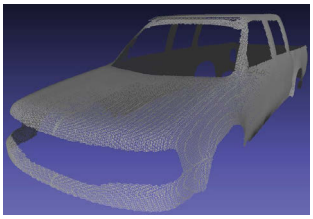
The assumptions in this thesis are outlined below:

- The AIRs can move (mounted on a mobile platform). Having an AIR base that can move is helpful for moving from one base location to the next. However, having the base move during the task execution (e.g. during the grit-blasting operation) may not necessarily be useful since most of the industrial robots have the necessary degrees of freedom (6 or more) to plan for a feasible motion within their workspace. Thus, for better operation, stability, and accuracy when performing the task, the AIRs' base is assumed to be fixed during the task execution.
- It is reasonable to assume that the AIRs can communicate and share information on the environment, their operation status, and their capabilities.
- The work in this thesis is mainly concerned with the complete coverage problem by multiple AIRs. The planning algorithm designed for dynamic environments mainly considers adaptability with respect to the coverage path (end-effector tool path) and assumes that a proper AIR motion planner is available.

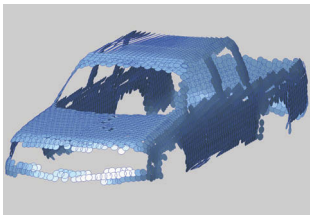
1.3 An Example Application

Grit-blasting the surfaces of objects [2, 3] is used as an example application in many of the case studies presented in this thesis. This application is similar to many other applications that require surface preparation, e.g. abrasive blasting, high-pressure cleaning, surface coating, and spray painting.

(a) Two AIRs operating on a vehicle.



(b) Point cloud of the vehicle.



(c) Target representation of the vehicle.

Fig. 1.3: Target representation of a vehicle's surfaces from a point cloud is used for two AIRs to perform a complete coverage task.

Grit-blasting is used to remove rust, old paint, or other debris from objects' surfaces which can be metallic, concrete, etc. The surfaces are cleaned by high-speed grit particles striking the surface. An AIR such as the one shown in Fig. 1.3a can be used to perform the grit-blasting operation. A grit-blasting AIR has been used in the iconic Sydney Harbour Bridge [18]. The AIRs utilized in the case studies of this thesis are comprised of an RGBD camera affixed to the end-effector of a 6 DOF Schunk industrial robot mounted on a mobile Neobotix MP700 base. Detailed specifications of the AIR are provided in Appendix A.

As shown in Fig. 1.3a, the end-effector of the AIR is equipped with a nozzle. The blasting stream exiting the nozzle aims at and follows the coverage path on the surface. The length of the blasting stream and the blasting angle with respect to the surface normal need to be within an acceptable range. Appendix A provides the values of the parameters related to the blasting stream and the end-effector nozzle which are found to be appropriate for the AIRs used.

Surface Representation: As mentioned previously, the information obtained from exploring and mapping an unstructured environment is used in the developed methodologies for multi-AIR complete coverage. In this thesis, since sensors are used to explore the environment, then a point cloud of the environment is generated (Fig. 1.3b). The point cloud is then used to decompose a surface into circular disks, called *targets* (Fig. 1.3c).

The method for generating the targets from a point cloud is from the references [7, 19, 20]. In brief, the method divides the environment into equally sized, cube-shaped volumetric pixels (voxels), and then associates 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. Principal Component Analysis (PCA) is used to find the target's normal vector.

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 end-effector speed and the properties of the tool attached to the AIR's end-effector need to be taken into account when determining targets' size and density.

If the dimensions of the objects are known (e.g. from a CAD model), then the AIRs can use the available model, but localization with respect to the objects would be needed. For an unknown object, exploration of the object by the AIRs is needed to build the map of the object. It can be convenient to use the point cloud from the sensors to decompose objects' surfaces into targets or grids. Note that targets are created on the surfaces only. 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.

1.4 Contributions

This thesis studies the research problem of multi-AIR complete coverage. A number of methodologies are developed to ultimately achieve optimal coverage of objects' surfaces. The contributions of this thesis are mainly related to Modules 3 and 4 of the flowchart shown in Fig. 1.2. More specifically, the main contributions are:

- **A Voronoi partitioning based approach for simultaneous area partitioning and allocation [21–23]:** A mathematical model is developed for simultaneously partitioning and allocating surface areas that can be reached by multiple AIRs at their current base placements. The approach utilizes Voronoi partitioning to partition objects' surfaces. Multi-objective optimization is conducted to allocate the partitioned areas to AIRs by optimizing the AIR team's objectives.

- **An optimization-based method to multi-AIR base placement for complete coverage [23–25]:** A mathematical model is developed for the problem of base placement such that the AIRs collectively cover the entire object while optimizing team's objectives and accounting for relevant constraints. This method enables the AIRs to find: (i) the minimal number of base placements for each AIR to operate from, (ii) the locations of the base placements for each AIR, and (iii) the visiting sequence of the base placements associated with each AIR. The model accounts for AIRs with different capabilities and objects with complex geometric shapes.

- **A stochastic optimization-based method to multi-AIR base placement for complete coverage under uncertainties [26]:** A mathematical model is developed that considers stochastic hybrid optimization for the problem of base placement under uncertainties. The presented model accounts for uncertainties in AIRs' base placements through Monte Carlo Simulations. The uncertainties are related to sensing and localization errors. Given probability distributions that represent the uncertainties in the AIRs' base placements, the AIRs select an optimal number (including locations), and sequence of base placements such that optimal complete coverage is achieved.

- **A prey-predator behavior-based algorithm for adaptive and efficient real-time coverage path planning in dynamic environments [27]:** Having assigned each AIR with surface areas at each base placement, complete coverage path planning is required to cover the allotted areas. An algorithm to adaptive complete coverage path planning is developed that accounts for dynamic environments and unexpected changes in the coverage areas of interest while aiming to achieve complete coverage with minimal cost. Examples of unexpected changes include new stationary or dynamic obstacles in the environment, and/or sudden change in the coverage area of interest (e.g. due to poor base localization). The algorithm is efficient for real-time implementation. The algorithm is inspired by the behaviors of animals in assessing fear with respect to predation risk and available resources.

1.5 Related Publications

1. M. Hassan, D. Liu, S. Huang, and G. Dissanayake. Task oriented area partitioning and allocation for optimal operation of multiple industrial robots in unstructured environments. In *13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1184–1189, Dec 2014. doi: 10.1109/ICARCV.2014.7064473

   This paper forms part of Chapter 3.

2. M. Hassan, D. Liu, G. Paul, and S. Huang. An approach to base placement for effective collaboration of multiple autonomous industrial robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3286–3291, May 2015. doi: 10.1109/ICRA.2015.7139652

   This paper forms part of Chapter 4.

3. M. Hassan, D. Liu, and G. Paul. Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2978–2984, Oct 2016. doi: 10.1109/IROS.2016.7759461

   This paper forms part of Chapter 5.

4. M. Hassan and D. Liu. Simultaneous area partitioning and allocation for complete coverage by multiple autonomous industrial robots. In *Autonomous Robots*, pages 1–20, 2017. doi: 10.1007/s10514-017-9631-3

   This paper forms part of Chapter 3.

5. M. Hassan and D. Liu. An approach to base placement and area partitioning for complete surface coverage by multiple autonomous industrial robots. In *4th International Doctoral Symposium on Mechanical Engineering at Hokkaido University (IDSHU)*, pages 89–94, Nov 2015

   This paper forms part of Chapters 3 and 4.

6. M. Hassan, D. Liu, and G. Paul. Collaboration of multiple autonomous industrial robots through optimal base placements. In *Journal of Intelligent and Robotic Systems*, Oct 2017. doi: 10.1007/s10846-017-0647-x

This paper forms part of Chapter 4.

7. M. Hassan and D. Liu. A prey-predator behavior based approach to adaptive coverage path planning. In *Transactions on Automation Science and Engineering (T-ASE), Conditionally accepted - Feb 2018*

   This paper forms part of Chapter 6.

8. M. Hassan, D. Liu, and D. Xu. A two-stage approach to optimization of multi-robot collaborative fiber placement. In *Journal of Intelligent and Robotic Systems (Submitted Jan 2018)*

1.6 Thesis Outline

This chapter of the thesis provided an introduction to the research problem, the motivation behind solving the problem, the scope of the work, and the main contributions of the thesis. The next chapter presents the literature related to the problem of complete coverage with particular focus on applying industrial robots. Chapters 3 to 6 present the methodologies developed for the multi-AIR complete coverage problem. Chapter 3 provides a graph-based approach for simultaneous area partitioning and allocation. Chapter 4 presents the optimization-based method to multi-AIR base placement for complete coverage. In chapter 5, the multi-AIR base placement problem is solved with uncertainties in base placements being considered by utilizing stochastic optimization and Monte Carlo Simulations. Chapter 6, presents a prey-predator behavior based algorithm for adaptive and efficient real-time coverage path planning. Case studies are provided in chapters 3 to 6 to demonstrate the effectiveness of the proposed methodologies. Concluding remarks, discussions, and future work are included in Chapter 7. The followings provide the detailed outline of chapters 2 to 7:

Chapter 2 first briefly explains the problem of complete coverage. It then presents some available research work on complete coverage that is relevant to a single robot operating on a flat surface, e.g. floor cleaning and harvesting robots. Then, multi-robot complete coverage problem is discussed and related work is presented. A significant portion of Chapter 2 is focused on discussing literature related to complete coverage of complex

objects by UAVs, AUVs, and industrial robots. Existing work for adaptive coverage to handle various uncertainties and conditions in an environment is also presented. Some literature related to the problems of base placement and area partitioning is also reviewed. Lastly, a conclusion on the related work is presented.

Chapter 3 studies the area partitioning and allocation problem. It starts with introducing and defining the problem. The requirements that need to be satisfied as part of solving the area partitioning and allocation problem are also presented. The proposed graph-based approach, which utilizes Voronoi partitioning, is then presented, and a mathematical model for optimal area partitioning and allocation is detailed. A standard multi-objective optimization algorithm, Nondominated Sorting Genetic Algorithm II (NSGA-II), is used to solve the optimization problem. Case studies are presented to demonstrate the effectiveness of the approach. The proposed approach is also compared to a pattern-based GA approach [29] which concurrently performs partitioning and coverage path planning.

Chapter 4 first introduces and defines the problem of multi-AIR base placement for complete coverage. The problem involves finding: (i) the minimal number of base placements for each AIR, (ii) the locations of the base placements for each AIR, and (iii) the visiting sequence of the base placements associated with each AIR. The method for solving the problem is first explained. Then a mathematical model that accounts for AIRs' team objectives and constraints is proposed. The optimization problem is solved using a standard multi-objective optimization algorithm, namely NSGA-II. An explanation of how NSGA-II is implemented for the problem under consideration is explained. Several simulations and real-world experiments are then carried out to demonstrate the effectiveness of the method.

Chapter 5 studies the multi-AIR base placement problem under uncertainties. The uncertainties in base placements can be due to sensing and localization errors. After introducing and defining the problem, the chapter provides a mathematical model. Due to the stochastic nature of the problem, the mathematical model utilizes Monte Carlo Simulations. Multi-objective stochastic optimization is used to solve the optimization problem, which is based on hybrid Genetic Algorithm (GA) and Simulated Annealing

(SA) approach. Case studies based on a real-world application are then conducted to demonstrate the effectiveness of the method.

Chapter 6 presents a prey-predator behavior based algorithm to coverage path planning for an AIR to cover the surface areas that it has been assigned. The proposed algorithm is designed such that it can handle unexpected changes in the environment, e.g. introduction of new stationary or dynamic obstacles. The algorithm is inspired by the behaviors of animals in assessing fear with respect to predation risk and available resources. After introducing the problem, Chapter 6 explains the prey-predator behavior. Then, the chapter presents the mathematical model which includes the development of a total reward function and optimization of two parameters within the total reward function. The total reward function with the optimized parameters guides the AIR in real-time to achieve complete coverage. Several case studies are then presented to show that the algorithm performs better than other adaptive CPP algorithms and that it enables an AIR to efficiently and effectively adapt its end-effector path to various changes.

Chapter 7 summarizes the problem and the key contributions. It then presents the limitations of the developed methodologies and suggests future work for overcoming the limitations and extending the developed methodologies to be applicable to a wider range of applications or scenarios.

Appendices provide: (A) the specifications and parameters related to the 6 DOF AIR used in the case studies, (B) the method used for calculating the torque value for each joint of an AIR, and (C) the method devised for obtaining feasible AIR poses for each target on a surface using a lookup table.

Chapter 2

Review of Related Work

The works in [5, 30] provide surveys on robotic complete coverage. There are many variations to the problem of complete coverage. Generally the variations can be categorized into two types: (i) complete coverage for applications where the end-effector of the robot must physically operate on all points or segments that represent the area of interest, e.g. floor cleaning [31, 32], field coverage [33–35] and grit-blasting [2]; and (ii) complete coverage for applications where it is not necessary to operate on all areas but rather to scan or inspect all areas of interest, e.g. the art gallery problem [36, 37] or the ship hall inspection problem [38]. In the former set of applications all points that represent the surfaces of objects must be visited or operated on. However, in the later set of applications, the problem is to select a subset of points or configurations such that by visiting them the whole area of interest (or all points representing the area of interest) can be observed or scanned.

Each application within each of the two aforementioned categories has a unique set of requirements. Hence, although there are algorithms that are aimed to be generic, most of the algorithms are designed for specific applications, conditions, or environments. For example, certain algorithms are efficient for coverage of flat surfaces and can be inefficient or impractical for coverage of objects with complex geometric shapes.

The literature review in this section first provides a brief overview of recent work on complete coverage of flat surfaces by a single robot. Then, multi-robots coverage is discussed,

followed by related work on coverage of complex objects or surfaces by UAVs and AUVs. Since the work in this thesis is mainly focused on AIRs, an overview of recent work on complete coverage of complex objects by industrial robots is presented. Then, related work on adaptive methods with respect to dynamic environments, unexpected changes, and uncertainties are provided. Literature related to base placement optimization, and area partitioning and allocation is also presented.

Note that as mentioned in Section 1.2, the scope of the work in this thesis focuses on the multi-AIR complete coverage problem. Other works, such as exploration, localization and mapping [7–9] and manipulator motion/trajectory planning [10–15, 17] are outside the scope of this thesis. Thus, the literature provided in this section does not cover these aspects in detail.

2.1 Complete Coverage of Flat Surfaces by a Single Robot

There are numerous algorithms available for a single robot to perform a complete coverage task on a flat surface [39, 40]. Example applications include floor cleaning [31, 32, 41], lawn-mowing and milling [42], and field coverage [33, 34, 43, 44]. The areas that these robots cover can be approximated as planar or flat surfaces. Galceran and Carreras [5] provided a survey on robotic complete coverage, and categorized the existing algorithms which include:

- Classical exact cellular decomposition: which includes trapezoidal and boustrophedon decomposition [45–47]. Trapezoidal decomposition is an off-line based method that decomposes the space into a number of trapezoidal cells allowing each trapezoid to be covered using a simple back-and-forth motion. Boustrophedon decomposition creates cells based on a subset of vertices of the obstacles, called critical points, instead of using all vertices of the obstacles. Thus, boustrophedon decomposition can create a fewer number of cells than the trapezoidal decomposition, and hence the overall path to cover the cells is shorter.
- Morse-based cellular decomposition [46]: which decomposes the space into cells with shapes decided by the Morse function chosen, and can be used on-line. Similar to

boustrophedon decomposition, the Morse-based cellular decomposition can handle obstacles that are non-polygonal.

- Landmark-based topological coverage [48]: which has the advantage of addressing a wider range of shapes such as the rectilinear and elliptical shaped surfaces. Unlike the Morse-based decomposition which can't handle rectilinear shaped surfaces, the topological coverage makes use of simpler landmarks to perform the decomposition referred to as the *slice decomposition*. That is, it uses a line to sweep through the space and then examines the behavior of the line (e.g. whether the line splits, merges, lengthens, or shortens) to construct the boundaries of the cells.

- Grid-based methods: which include the wavefront [49], the spanning tree [50], and the neural network [51, 52] based algorithms. These algorithms represent the surface as a set of uniform grids where each grid can be free or occupied by an obstacle. Then the obstacle-free grids are covered based on a set of rules or equations that direct the robot from one grid to the next. For example, in the wavefront algorithm, the surrounding grids of the goal grid are given a value of 1, then the grids surrounding those labeled as 1 are given a value of 2, and so on. At each step, the robot covers a neighboring unvisited grid that has a higher value.

2.2 Complete Coverage by Multiple Robots

Recently, research is being focused on multi-robot coverage for different environments, conditions, and applications. Examples include cleaning a large public space [53], sustainable broad-acre agriculture [54], complete coverage while considering energy consumption [55, 56], coverage of non-convex shaped surfaces and in non-Euclidean metric spaces [57], and repeated multi-agent coverage for surveillance and target detection [58]. Below are explanations of some of the interesting multi-robot complete coverage approaches.

Bio-inspired approaches for the multi-robot complete coverage problem have been developed [5, 59, 60], which are based on the behaviors found and studied in nature. An approach is presented by Ranjbar-Sahraei et al. [60] where the behavior of ants is used to prevent other robots from entering an area apportioned by a robot. The robots move

in a circular motion on a planar surface, 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, it avoids the intersection of another robot's border. As a result, the robots gradually spread out while preventing intersection with other robots' borders. The paper Ranjbar-Sahraei et al. [60] 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 of the robots when an intruder is detected, by decreasing the territory areas of the robots.

Fazli et al. [61] developed algorithms for the problem of multi-robot repeated area coverage. In brief, the overall approach is as follows: (i) generating a number of points, termed static guards, on the surface such that the whole environment can be jointly observed by the robots if it is assumed that there are as many robots as there are guards, (ii) creating a graph by appropriately joining the guards and the nodes of the workspace, (iii) reducing the size of the graph, and (iv) 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 path by going through all static guards and then appropriately allocating a portion of the path to each robot). For each of the steps mentioned above, the authors present one or more algorithms to address the individual problem.

An algorithm that falls in the category of sensor-based coverage of rectilinear surfaces is presented in [62]. In this work, the surface is represented as a set of disks. The mobile robots pass through the centroid of the disks to cover the surface. To determine the visiting order of the disks by a robot, the initial locations of the robots are 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 [63] uses exact cellular decomposition and flow network methods to achieve complete coverage using a single or multiple mobile robots. In this work, using a sweeping line, the free space of the surface is first partitioned into many 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.

The work in [64] 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 used 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 in order to obtain a full scan.

Zheng et al. [65] 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 terrain that has non-uniform traversability. Comparisons 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.

2.3 Complete Coverage Using UAVs and AUVs

In many of the existing methods, the developed algorithms are efficient for planar surfaces. Algorithms that are designed for coverage of complex surfaces are mainly applied to Unmanned Aerial Vehicles (UAVs) [66–69], Autonomous Underwater Vehicles (AUVs)

[70, 71], or industrial robots [72–74]. To provide a brief clarification on how these algorithms differ, literature related to complete coverage of complex objects by UAVs and AUVs is first provided followed by literature that is specific to industrial robots.

Danner and Kavraki [75] 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 of an area 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 [5]. The works in [38, 70] 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.

Maza and Ollero [76] have used convex decomposition to divide the search environment into a number of convex polygons. Each robot is then responsible for covering a convex polygon using a zigzag pattern. They present algorithms that enable multiple heterogeneous UAVs to cooperate with each other and find an object in the search environment.

The work in [77] takes into account the energy consumption of UAVs as well as requirements such as coverage and sensor resolution to compute a path. The authors point out that many of the existing works focus on geometric constraints and don't account for robot-related constraints such as sensor resolution, maximum speed, and weight. The work presents an energy model which is used to calculate the energy consumption of a path and in turn determine the speed which minimizes the energy consumption. The energy model can also examine whether an area of interest can be covered with the available amount of energy.

An on-line and sensor-driven coverage approach for AUVs with sidescan sonar sensors is presented in [78]. The application is seabed coverage for mine countermeasure. The

approach makes use of concepts such as information theory, hexagonal cell decomposition, graph generation (directed acyclic graph), and computation of branch entropy. The approach has various advantages which include generating shorter path length and time, accounting for sensor performance and environmental factors, avoiding the need for off-line computation of waypoints, and capability of covering complex and non-convex surfaces.

The field of view of the sensor attached to an AUV can change depending on the distance of the AUV from the seabed. Hence, the work by Galceran and Carreras [79] considers operating at different depths to obtain consistency in a sensor's field of view. In doing so, the surface of interest is segmented into regions with similar depths. It is then easier to determine the best sweep orientation and the inter-lap spacing of the path for each region. The result is a better-optimized path length and coverage efficiency. In another work, Galceran et al. [80] presented a method for coverage of projectively planar surfaces using a bathymetric map information. A Morse-based boustrophedon decomposition algorithm is used for coverage path planning of effectively planar surfaces, whereas for high-slope regions a slicing algorithm is used where horizontal planes (offset by a distance) slice through the high-slope regions and create a path at the intersections with the surface.

The work by Lee et al. [81] introduces the concept of "artificial island" for AUV map building of underwater terrain. A terrain is considered as a set of planes at different depths. The AUV starts at a location in a plane and covers the plane while avoiding the islands. Coverage is based on zigzag motion, following parallel lines, and divergence to a cape point of an island or inlet. The AUV then moves to the next plane and uses the information collected in the previous plane to shorten the path. In a more recent work [82], the artificial island-based coverage is used with the spiral path terrain coverage algorithm where the surface of a steep terrain is covered using a three-dimensional vector. A hybrid decision module is used to select, and switch between, the appropriate coverage algorithm based on the slope of the surface.

2.4 Complete Coverage Using Industrial Robots

In some environments, it may be acceptable to assume that the surfaces an AUV or UAV covers are planar, or that the surfaces contain high-slope regions and effectively planar regions and hence can be segmented. In certain conditions, it is sufficient for AUVs or UAVs to plan a path on a plane [83] (i.e. to move at a constant depth or height) while covering non-planar surfaces, because physical contact with the surface may not be necessary for sensing. However, an industrial robot may need to move its end-effector point continuously in 3D and with different orientation at each time step while operating on a complex object. Hence, generating the end-effector path as well as the manipulator path of an industrial robot is a complex problem. Moreover, if the object has a complex geometric shape, then it may not be useful or possible to segment the surfaces into planar and non-planar regions.

Research in CPP for industrial robots has largely focused on the application of spray painting, and particularly in the automobile industry [73, 74, 84–86]. Due to the nature of this task, planning time prior to the spray painting process is not critical as the preplanned path is repeated on a large number of vehicles. In such applications, precise dimensions of the object are usually available [74], e.g. through CAD models. Having accurate dimensions of the objects can simplify many aspects of the CPP problem. For example, segmentation can be done based on the topology or surface normals [74, 87], i.e. complex surfaces are segmented into patches that are simple, which in turn enables certain CPP algorithms to be implemented.

Another example of CPP for an industrial spray painting robot is presented in [88] where regular surfaces are considered; namely, planes and cylindrical, conical and spherical surfaces. The work aims for consistent coating thickness using an optimization model that generates the optimal path parameters (i.e. spraying gun velocity and stroke distance). The approach can also be applied to a complex free surface with a large curvature that can be segmented into regular surfaces.

Note that the above approaches may only be practical for repetitive tasks and off-line implementation, and they have their limitations, e.g. may only be applied to regular surfaces,

or require segmentation. Segmentation of the surfaces into smaller and approximately planar regions may not produce good results (e.g. may divide the surface into many small patches, and hence cause many discontinuous trajectories for an industrial robot), or may be infeasible (e.g. due to the complexity of the surface curvatures).

2.5 Complete Coverage Using Autonomous Industrial Robots

There has been limited research on autonomous industrial robots performing CPP in unstructured environments. An example of an application is grit-blasting of steel surfaces [7]. In [13] and [89], trajectory planning and tool path planning are combined, and a Genetic Algorithm based method is employed to optimize the planning. In [90], closed and orientable surfaces that are embedded in $\mathbb{R}^3$ are covered on-line by using a modified version of the Morse decomposition method that can manage non-planar surfaces.

For autonomous industrial robots operating in unknown and unstructured environments, exploration and mapping may be necessary [7]. Many grid-based methods for complete coverage decompose the surface into uniform grid cells [5]. The reason is that performing segmentation from a point cloud can be challenging. Hence, researchers have focused on converting point cloud data to other forms of representing the surfaces, e.g. using scale-like disks [19].

Another application is fiber placement for large and complex structures (e.g. manufacturing airplane wings). One of the important tasks in Robotic Fiber Placement (RFP) is generating the tool paths that the robot's end-effector (the roller) follows, and by doing so, places the fiber on the surface. A composite structure can be made up of many layers of fiber, and each layer consists of many tool paths. When generating the tool paths, information such as the geometry of the object, the width of the fibers, the number of layers and the acceptable overlap or gap between fibers need to be taken into account [91] so as to achieve optimal complete coverage. The problem of generating tool paths has been widely studied [91].

Researchers have focused on different aspects of the tool path planning problem so as to achieve optimal coverage (without gaps or unwanted overlaps). As part of the overall

method in [92], the authors aim to smooth the path, i.e. to determine and smooth the orientation of the tool head along the path whilst meeting the required quality and constraints. Shirinzadeh et al. [93] applied their developed SCAR path planning algorithm to open-contoured structures by formulating a set of surface curves that represent the tool paths. The algorithm aims to create uniform layers of fiber with no gap or overlap between tows (fiber stripes). However, as Yan et al. [94] argue, many of the studies assume a constant value when offsetting a reference tool path to generate a new tool path. Thus, Yan et al. [94] consider a varied offset and other improvements to provide a more accurate approach to path generation. These improvements are made to prevent gaps or overlaps between successive tows which can cause incomplete or uneven coverage.

There are a very limited number of studies on collaborative fiber placement machines. These studies are mainly focused on developing systems that are coupled together to provide various advantages, and only consider one robot to lay the fiber. The other components of the coupled system mainly help with carrying the mandrel, turning the mandrel, assisting with reachability of the roller, etc. For example, in [95], a collaborative machine is presented which comprises a 6-DOF manipulator, a 6-RSS parallel platform, and a spindle to hold the mandrel. They construct the kinematics of the overall machine, analyze the workspace and prove that the overall machine can enlarge the workspace, simplify trajectory planning and in-turn improve productivity. Another example of an RFP machine is presented in [92], where the authors take advantage of the additional degree of redundancy to decrease the kinematic loads of the control joints.

2.6 Adaptive Coverage Problem with Respect to Change and Uncertainties

It may be necessary for a complete coverage algorithm to have some degree of adaptability to overcome aspects such as uncertainties, unexpected changes in the environment, and dynamic obstacles. Sources of uncertainties can be sensing and localization errors; and errors in predicting certain parameters, e.g. the speed of a robot and the trajectory of a dynamic obstacle. Examples of unexpected change in an environment include a sudden

introduction of unknown and stationary obstacles, sudden modification to the coverage area of interest, and the presence of dynamic obstacles with unknown trajectory.

The adaptive behavior that a complete coverage algorithm needs to have is dependent on the application and the type of robots used, and may be required for both off-line and on-line implementations. For instance, the base placement of an AIR may be subject to uncertainties due to localization and sensing errors. Hence, it can be beneficial for a team of AIRs to plan their base placements such that uncertainties are accounted for.

Many of the existing on-line or real-time algorithms assume only stationary obstacles being present in the environment. In [96] a multi-robot coverage algorithm that is based on spanning tree is presented, which can be used for on-line implementation. The algorithm focuses on minimizing the overall completion time of the task and is proven to be robust. In [97], an on-line coverage algorithm is presented for coverage of planar surfaces, which utilizes boustrophedon motion. The algorithm can be used for a team of robots and unknown environments. Another on-line algorithm is presented in [98] where A* search algorithm is combined with the boustrophedon motion to achieve complete coverage. A sensor-based on-line coverage path planning algorithm is presented in [99] where a spiral filling rule is used, which is tested for planar surfaces. A family of problems is concerned with the limited resources of the robots [100, 101]. In [100], an algorithm is developed that takes into account the battery capacity of the mobile robot when planning for complete coverage of planar surfaces in unknown environments. For tethered mobile robots, the length of the cable and avoidance of cable crossing by the robot should be taken into account [101]. These algorithms are tested on planar surfaces and their effectiveness with respect to dynamic obstacles is not studied.

There is a growing research focusing on developing complete coverage algorithms that are adaptive and can account for various changes and uncertainties related to the application under consideration. The work in [102] presents a probabilistic approach to high-confidence cleaning with the aim of achieving a dirt level that is below a user-defined threshold on all areas. A floor cleaning robot may not achieve the same standard of cleanliness throughout the floor, due reasons such as unequal distribution of dirt, failure of the cleaning unit to pick up all the dirt at a location, and uncertainties in motion and sensing. Thus, the

approach in [102] estimates dirt distribution and trajectory of the robot, and using the developed probabilistic models for the cleaning unit and the dirt sensor enables complete coverage with a low level of dirt throughout the floor. Hence, the approach is adaptive in that it can re-plan cleaning path so as to achieve a high-confidence cleanliness of dirty areas.

In the work by Bretl and Hutchinson [103], the uncertainty in mobile robot's position and velocity is taken into account and an approach for guaranteed coverage is proposed. However, due to over-coverage a longer path is generated. Instead of a probabilistic model, a worst-case model of uncertainty is considered which guarantees complete coverage. A useful application for the approach is landmine detection and removal which requires guaranteed complete coverage to prevent human casualties or injuries. Another work [104] studies uncertainty in sensing and actuation for complete coverage and presents a performance measure that when applied to a policy for coverage, the probability associated with covering a certain fraction of the surface can be obtained.

Galceran et al. [80] account for pose uncertainty of an AUV that is surveying or inspecting the seabed. They present a two-stage approach: (i) off-line planning where the complete coverage path of the AUV is calculated, and (ii) adaptive real-time re-planning where periodically part of the path is reshaped on-line using the STOMP algorithm by taking into account the new information obtained from the range-sensing sonars. The adaptive behavior of the approach is mainly to minimize the distance from the executed path to the nominal path.

Another approach that accounts for pose uncertainty of an AUV is presented in [105]. In this approach, the pose estimate is combined with the sensor model and a probabilistic representation of coverage is obtained. As part of the approach, a path planning framework that uses information theory is also proposed. The overall framework can then help generate paths that can guarantee coverage according to probabilistic coverage criteria.

Neural network-based coverage fits in the grid-based coverage category and has the advantage of being adaptive [5] in that it can re-plan the path when it meets a deadlock situation or when a stationary or dynamic obstacle is introduced into the environment.

In neural network-based approaches, the surface is represented as a set of uniformly distributed neurons. The neural activity of the neurons that are in collision with obstacles is negative, whereas the neural activity of the neurons that are in the uncovered areas of the surface are positive. The positive neural activities propagate to the state space and globally attract the robot. However, for large surfaces such as the seabed, a 2D representation is unrealistic, and the neural network approach can be computationally intractable [5]. The work in [29] tries to reduce the computational burden of the approach by adding rolling path planning and heuristic search to neural network-based approach. The work shows that using the combined method, shorter path length with fewer turns and repeated coverage can be obtained.

2.7 Base Placement and Area Partitioning for Complete Coverage

For large objects or objects with complex geometric shape, autonomous industrial robots need to operate from a number of base locations so as to achieve complete coverage of all surfaces. The task of determining appropriate base locations for the AIRs is termed as the base placement problem. In addition to the base placement problem, the autonomous industrial robots need to appropriately partition and allocate the surface areas of the objects amongst themselves, particularly since the workspace of the AIRs at certain base locations may overlap. In doing so, they need to prevent repeated coverage or missed-coverage. In this section, a brief overview of existing work related to these aspects is presented.

2.7.1 Base Placement Optimization

Appropriate base placement of industrial robots enables time-efficient and effective performance by the robots. There is some literature available for finding an appropriate base placement for a single robot in different environments, such as in manufacturing environments [106, 107] and underwater environments [108, 109].

In [106] numerical approaches are developed to estimate the workspace of an industrial robot and to find a base placement for the robot. Genetic Algorithm is used to optimize the base placement of an industrial robot with the objective of containing a set of critical point (from the work cell) within the workspace of the robot. The base placement optimization is performed on seven different industrial robots to check the suitability of each robot for a given work cell. In [107] the workspace of an open-loop robotic manipulator is impelled towards a set of critical points using a cost function, while constraints are applied to ensure reachability of the end-effector to the critical point. An advantage of the approach is that inverse kinematic doesn't need to be solved for each critical point when finding a base placement.

The problem of finding an optimal base placement for a robot can be computationally expensive due to the size of the search space and hence, researchers often simplify the problem by considering only a limited number of critical discrete end-effector positions [110] when optimizing the base placements. Mitsi et al. [110] developed an optimization based method that combines Genetic Algorithm, quasi-Newton algorithm, and constraints handling methods to find the optimal placement of a single manipulator by considering discrete end-effector positions.

Many of the methods utilize optimization techniques. The objectives taken into account for the optimization are specific to the task. Vosniakos and Matsas [111] presented an approach that uses Genetic Algorithm and considers maximizing manipulability of a single manipulator to deal with the high accuracy required for the milling application. However, other performance measures such as the task-dependent and direction-selective performance indexes [112] are also shown to be effective under certain conditions when finding an optimal base placement for a robot.

Note that in the above literature, only a single base placement is considered for a single industrial robot. The problem becomes increasingly complicated when (i) multiple robots are involved in carrying out the intended task, (ii) each robot must find multiple base placements (and their visiting sequence) such that the robot team can collectively complete the overall task with minimal cost, (iii) a large number of points need to be covered instead of a few critical points, and (iv) multiple objective need to be considered.

2.7.2 Area Partitioning

For multi-robot coverage, it may be necessary to partition and allocate certain areas of interest amongst the robots. An example scenario where area partitioning and allocation is required is when the workspace of robots overlap and as a result, more than one robot can reach these surface areas.

Graph-based partitioning approaches [113] have been widely used in different fields as a mean to partition objects or surfaces into smaller pieces. Many researchers have used or studied convex decomposition [114–117] to divide an arbitrary shape into a number of convex or near-convex polygons. For example, Lien and Amato [118] proposed a method for decomposing a polygon which contains holes, where holes can represent areas that are not of interest or include an obstacle. Another graph-based approach that can be used for area partitioning is Voronoi diagram [119, 120]. Voronoi partitioning can divide a graph into n Voronoi cells based on the location of n seed points. The size and shape of each cell are affected by the location of the seed points.

In robotic research, convex polygon decomposition is used for multiple-robot workspace division. For example, a cooperative search method by multiple UAVs in an unknown environment which incorporates convex polygon decomposition is presented in [76]. Hert and Lumelsky [121] studied the terrain covering problem by multiple robots. In their paper, terrain covering capabilities of the robots are considered when dividing the area into n polygonal sections. Each polygonal section is visited once by one of the n available robots.

Another method for partitioning an area is grid graph bisection method [122–124]. Using this method, the aim is to partition a two-dimensional grid graph into two cells that are of equal size. The cells can then be partitioned again into two smaller cells, and so on. It is preferred to partition the graph into two cells such that the number of edges connecting the cells is minimal. There are different variations of the grid graph bisection method. The work in [124] aimed to reduce the computation complexity to $O(n^4)$ algorithm for solid grid graphs (with no hole). The work in [123] presents an exact algorithm based on the branch-and-bound framework, which is, unlike many other approaches, fully combinatorial.

A partitioning method called Virtual Door algorithm is presented in [31], which considers partitioning a large area into numerous smaller rectangular shaped sections. This Virtual Door algorithm is shown to achieve time-efficient complete coverage by multiple cleaning robots. This method partition the areas of interest mainly to carry out the task in a time-efficient manner by mobile robots.

2.8 Conclusion

Most of the existing complete coverage algorithms are application-specific and are effective for a certain condition or environment. Lots of research has focused on coverage of simple planar surfaces for applications such as floor cleaning and harvesting. Research work has recently shifted to handle more complicated scenarios where objects can have complex geometric shapes or the environment is complex, e.g. in applications where AUVs or UAVs are utilized. There is also a trend towards building algorithms with some degree of adaptability for a specific application. However, there has been a very limited attention towards multi-AIR complete coverage problem. The use of an AIR for tasks such as spray painting and grit-blasting imposes its unique set of requirements and constraints. Most industrial robots are still largely deployed in factory settings where the environment is purpose-built for a repetitive task, and autonomous operation is limited. The need for complete coverage by a team of AIRs adds further complications to the problem. This is particularly true when the AIRs have to account for certain uncertainties, be adaptive to perform a complete coverage task efficiently in environments where unexpected changes can occur, and simultaneously aim to achieve optimal outcomes. Thus, the work in this thesis focuses on developing methods and algorithms for a team of AIRs to perform a complete coverage task for large and/or complex objects in dynamic environments. The developed methodologies in this thesis have 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 are necessary.

Chapter 3

A Voronoi Partitioning Based Approach for Simultaneous Area Partitioning and Allocation

For complete coverage of surfaces by multiple AIRs, it is important for the AIRs to appropriately partition the surface areas of an object and allocate the partitioned sections amongst themselves. Given the base locations of the AIRs, each AIR will be able to cover a certain part of the objects' surface areas. Some of these surface areas may be covered by two or more AIRs. These areas are termed as overlapped areas. Partitioning and allocation is carried out on the overlapped areas where the AIRs share their workspace. An example of overlapped areas is shown in Fig. 3.1. An approach to the above problem is presented in this chapter, which takes into account objects with complex geometric shapes, and AIRs with different capabilities (e.g. different tool coverage size or end-effector speed). 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. The approach is validated using eight case studies[1] that consist of comparative studies, complex simulated scenarios as well as real-world scenarios using data obtained from real objects and applications.

The developed Area Partitioning and Allocation (APA) approach utilizes Voronoi partitioning (also referred to as Voronoi Tessellation, Voronoi graph, and Voronoi diagram) to partition objects' surfaces. Voronoi partitioning creates Voronoi cells on the surface areas of an object (or multiple objects). For n AIRs, n Voronoi cells are created. The size and

[1] A video for Case Study 8 can be viewed at $https://youtu.be/1TjrcXKpy4$.

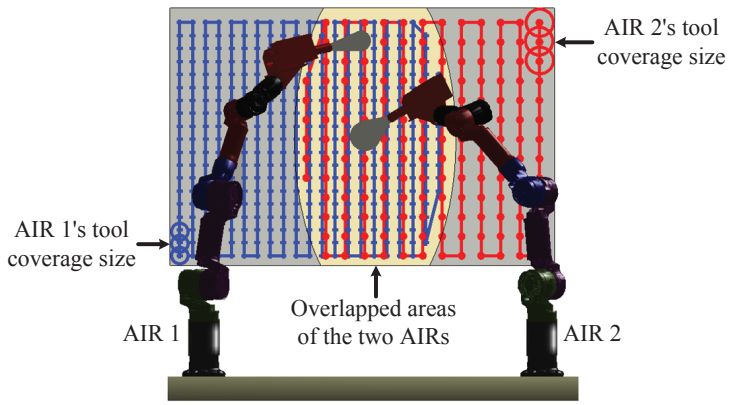


Fig. 3.1: The overlapped areas of two AIRs that have different tool coverage size are shown.

shape of each cell are iteratively modified using a multi-objective optimization algorithm. The aim is to create cells that will result in a minimal cost.

Multi-objective optimization is used to allocate the partitioned areas to the AIRs whilst optimizing AIR team's objectives. In addition to minimizing the overall completion time and achieving complete coverage, manipulability measure and joints' torque are also optimized. As to be discussed in Section 3.2.2 (Mathematical Modeling), these objectives are in conflict with each other. That is, optimizing one of the objectives can only be done at a cost to another objective (or subset of objectives). Hence, multi-objective optimization is suitable when conflicting objectives are present. This work is the first to formulate the problem of simultaneous area partitioning and allocation into multi-objective optimization while considering conflicting objectives and constraints related to multiple AIRs performing complete coverage tasks.

Unlike most of the existing literature on complete coverage, the presented approach focuses on simultaneous area partitioning and allocation rather than Coverage Path Planning (CPP). A CPP algorithm, such as the algorithm presented in Chapter 6, can then be applied to plan the end-effector path for the allocated area of each AIR. As discussed in Chapter 2, many of the existing methods, e.g. in [60–63], are shown to be efficient for planar or projectively planar (2.5D) surfaces. The presented APA approach in this chapter considers complete coverage of surfaces that may be non-planar, complex in shape and separated (unconnected) from each other. Some of the algorithms are designed for specific

applications; for example in [62] the approach has only been validated using rectilinear surfaces and rectilinear moves (90° and 180° robot turns). The APA approach does not have these limitations. Additionally, the APA approach has the advantage of being able to use the point cloud information that is generated from sensing the environment.

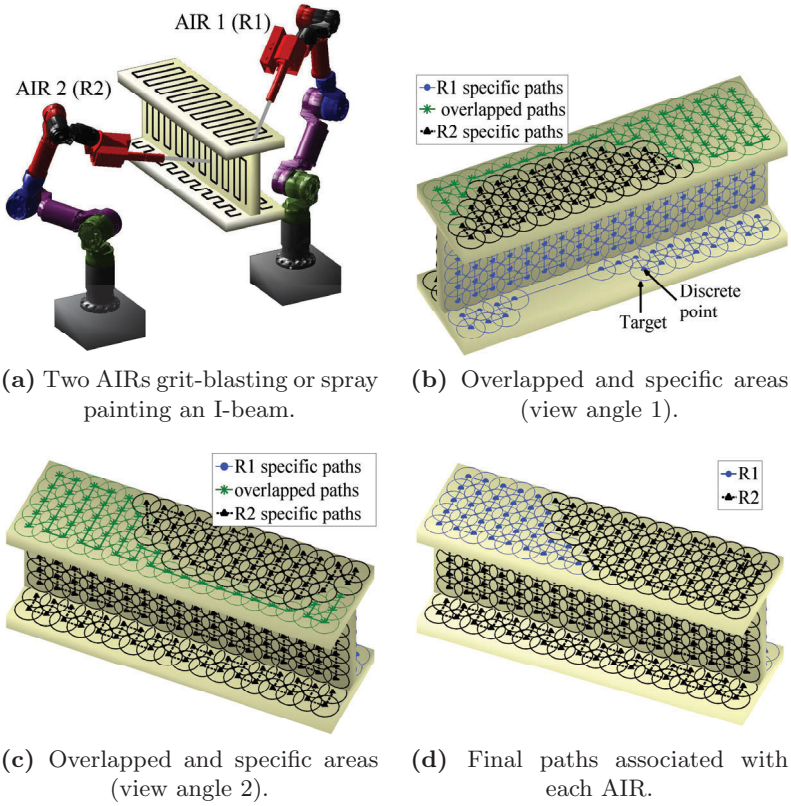
The main results of this chapter were previously included in the following papers: Hassan et al. [21], Hassan and Liu [22] and Hassan and Liu [23].

The rest of the chapter is organized as follows. Section 3.1 presents a detailed explanation of the problem and the requirements that need to be satisfied. Section 3.2 presents the methodology, consisting of two sub-sections: the APA approach is presented in Section 3.2.1 and the mathematical model is detailed in Section 3.2.2. Case studies are presented in Section 3.3 to demonstrate the effectiveness of the approach for different scenarios and conditions. Discussion is then provided in Section 3.4. Finally, Section 3.5 concludes the work presented in this chapter.

3.1 Problem Definition

As an example scenario, consider the I-beam object in Fig. 3.2a where the I-beam needs to be grit-blasted or spray painted. In this example, two identical AIRs are deployed. However, AIRs may have different capabilities and their base placement 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 Figs. 3.2b and 3.2c. Thus, the problem is to partition and allocate the overlapped areas amongst the AIRs such that AIR team's objectives are optimized.

As illustrated in Fig. 3.2b, a *target* is a circular disk created around a discrete point, such that the discrete point is in the center of the target. The definition of targets and the method to create targets were explained in detail in Section 1.3. Figure 3.3 is another example showing the overlapped areas of three AIRs that have different capabilities. AIR



(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. 3.2: 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.

3 is equipped with a tool that can, at each step, cover larger areas than the other two AIRs. Hence, the targets for this AIR are larger in size. Similarly, the second AIR's tool coverage is larger than the first AIR's tool coverage.

Let a_{ij}^t be the surface area covered by the target $\boldsymbol{o}_{ij} \in O_i$ where i is the AIR index (ith AIR) and j is the target index (jth target), 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, $\boldsymbol{q}_{ij}^f = [\theta_1, \theta_2, \ldots, \theta_{n^J}]$, needs to be found (Appendix C) that can reach the target $\boldsymbol{o}_{ij}$ 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^J 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.

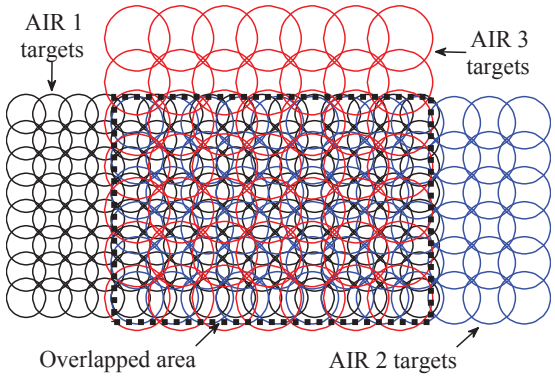


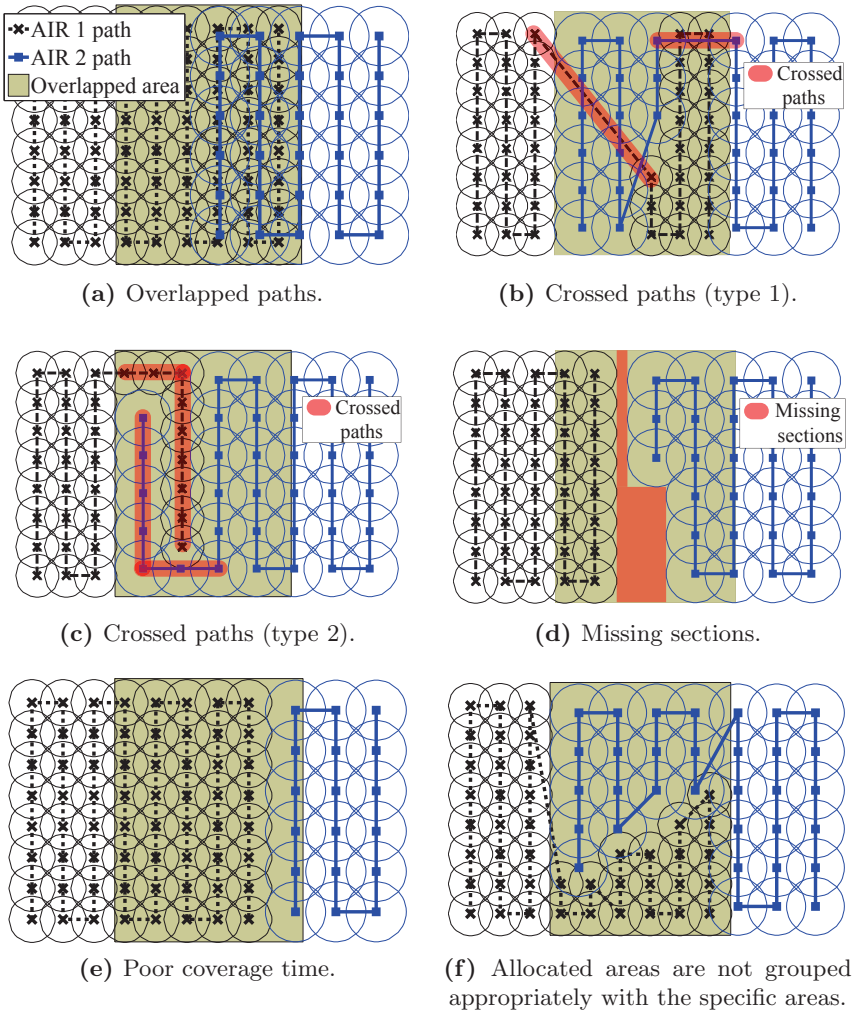
Fig. 3.3: Three AIRs with different capabilities, each associated with a different set of targets, are used to grit-blast a flat plate.

The aim of the APA approach is to solve the problem of partitioning and allocating the overlapped areas such that optimal complete coverage of the object's surfaces is achieved. 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 first example, it would be expected that the output of the APA approach will be used to modify the paths shown in Figs. 3.2b and 3.2c such that for each AIR, the path from the allocated areas is smoothly continued to the specific areas as shown in Fig. 3.2d.

The APA approach is developed while taking the following requirements into consideration:

1. Prevent overlapped paths: Let A represent the overlapped areas. The aim here is to have $A_i \cap A_{\acute{i}} = \emptyset$ for $\acute{i} = 1, 2, \ldots, n, i \neq \acute{i}$ where $A_i \subseteq A$ is the area allocated to the ith AIR. Unlike the condition that is shown in Fig. 3.4a, the APA approach prevents 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 Figs. 3.4b and 3.4c. In type 1 shown in Fig. 3.4b, the paths of the two AIRs cross each other, which may cause a higher chance of collision. In type 2 shown in Fig. 3.4c, the crossed paths may prevent the AIRs from maneuvering more freely during the task execution since the manipulators of the AIRs will be



(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.

Fig. 3.4: Examples of unacceptable solutions where in each example at least one requirement is not met.

operating in close proximity to each other in these regions. Let $A_{i,k}$ and $A_{i,\acute{k}}$ be two areas allocated to the ith AIR. The aim is to have $A_{i,k}$ and $A_{i,\acute{k}}$ be as close to each other as possible.

3. Prevent missing sections: The aim here is to have $\bigcup_{i=1}^{n} A_i = A$ so as to prevent missing section. In Fig. 3.4d, some sections of the overlapped areas are not covered by any AIR, that is, *missing sections* are created. The APA approach does not cause any missing sections.

4. Minimize the overall completion time of the task: The APA approach minimizes the makespan or in other words, the overall completion time of the task. Thus, the

aim is to $\min_{A_i,\forall i} \mathrm{T}(A_1, A_2, \ldots, A_n)$ where $\mathrm{T}(\cdot)$ calculates the makespan based on the allocated areas of the AIRs. As an example, Fig. 3.4e shows a scenario where poor makespan is obtained due to poor allocation of the overlapped areas to the AIRs.

5. Improve the grouping of the targets: Unlike the condition that is shown in Fig. 3.4f, the targets O_i^{al} in the allocated areas of the ith AIR should be closely grouped with the targets O_i^s 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 motion of the AIR is less constricted by other AIRs, and the likelihood of collision is reduced.

6. Maximize manipulability measure: The APA approach is capable of optimizing the performance of the manipulators attached to the AIRs. As an 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. Thus, the aim is to $\max_{A_i,\forall i} \mathrm{W}(A_1, A_2, \ldots, A_n)$ where $\mathrm{W}(\cdot)$ calculates the sum of manipulability based on the allocated areas of the AIRs.

7. Minimize joints' torque: Another performance measure is the minimization of the torque experienced by the AIRs' joints during the task execution, which the APA approach takes into account. The aim is to $\min_{A_i,\forall i} \mathcal{T}(A_1, A_2, \ldots, A_n)$ where $\mathcal{T}(\cdot)$ calculates the sum of torques experienced by the AIRs' joints to operate on the allocated areas.

3.2 Methodology

3.2.1 The APA Approach

Algorithm 3.1 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 surfaces 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. 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 $\boldsymbol{o}_{ij}$ with appropriate end-effector position and orientation needs to be found, e.g. using the lookup table explained in Appendix C. The first step of the approach is to find the overlapped areas of each AIR, and line 1 of Algorithm 3.1 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 [119] 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 Voronoi graph's seed points, as will be explained in more details in Section 3.2.2.1. Thus, as shown in line 2 of Algorithm 3.1, 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 to 3 mentioned in Section 3.1, i.e. it can prevent overlapped paths, crossed paths and missing

Algorithm 3.1 Area Partitioning and Allocation (APA).

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

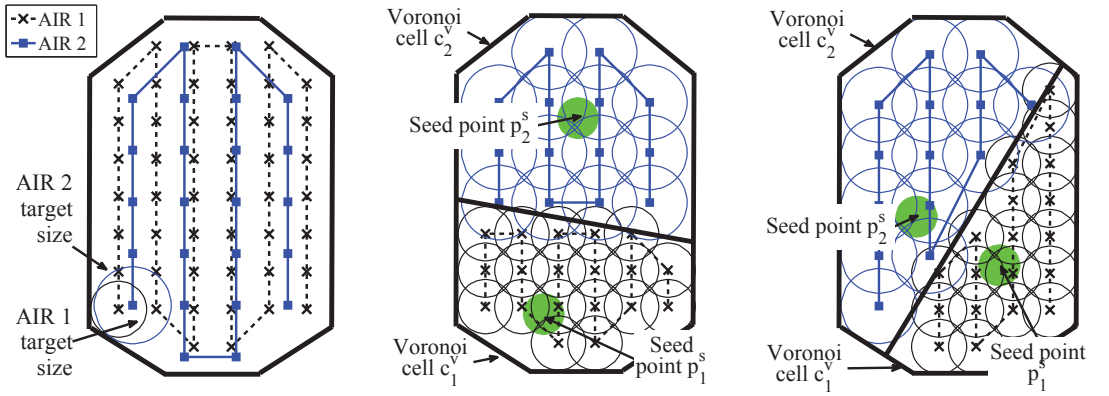
sections. There is, however, a special condition where Voronoi partitioning may fail to prevent missing sections when more than two AIRs are deployed. This issue is discussed in Section 3.3.6 and a solution to the problem is presented.

The advantage of using multi-objective optimization [128] is that multiple 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 manipulators 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 model (Section 3.2.2.2). Line 3 of Algorithm 3.1 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 $\boldsymbol{y}^f$ from the Pareto front (line 4 of Algorithm 3.1) is dependent on the priority of the objectives determined based on the application being considered. A selection strategy for selecting $\boldsymbol{y}^f$ is explained for a particular application in Section 3.3.5.

3.2.2 Mathematical Model

3.2.2.1 Design Variables

The position coordinates of the seed points of the Voronoi graph 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 belongs to the ith AIR and is associated with a seed point position $\boldsymbol{p}_i^s$. Figure 3.5a illustrates overlapped areas and overlapped paths of two AIRs. The overlapped areas can be partitioned using Voronoi partitioning as shown in Figs. 3.5b and 3.5c. It can be seen that the targets in each Voronoi cell are closest to the seed point of the cell; therefore, $O_i^{al} = \left\{ \boldsymbol{o} \in O_i \;\middle|\; \|\boldsymbol{o} - \boldsymbol{p}_i^s\| \leq \|\boldsymbol{o} - \boldsymbol{p}_{\acute{i}}^s\|,\ \forall \acute{i} \in \{1, 2, \ldots, n\} \right\}$ where O_i^{al} is the set containing the allocated target of the ith AIR and O_i is the set containing the targets representing the overlapped areas of the ith AIR. Note that the overlapped areas to be covered by the AIRs is fixed and Voronoi partitioning simply partitions this fixed area into n cells. Changing the position of the seed points changes the shape of the cells; however, the total area



(a) Overlapped area of two AIRs. **(b)** Voronoi partitioning example 1. **(c)** Voronoi partitioning example 2.

Fig. 3.5: Two examples of Voronoi partitioning where the overlapped areas of two AIRs are partitioned based on the locations of the seed points.

covered by the cells remains fixed. Thus, an advantage of utilizing Voronoi partitioning is that an additional objective function is not needed to maximize coverage.

Thus, the design variables are

$$Z = \{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \ldots, \boldsymbol{p}_n^s\}, \tag{3.1}$$

as shown in line 2 of Algorithm 3.1. The goal is to obtain seed point positions and create Voronoi cells that will optimize AIR team's objectives.

3.2.2.2 Objective Functions

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

Objective 1 - Minimal Makespan: 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:

$$\min_{Z} \mathrm{F}_1(Z) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathrm{T}_i(Z) - \bar{t} \right)^2 \tag{3.2}$$

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

$$\mathrm{T}_i(Z) = \frac{\mathrm{L}_i^o(Z) + l_i^s}{v_i} \tag{3.3}$$

where $\mathrm{L}_i^o(Z)$ and l_i^s are 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,

$$\mathrm{L}_i^o(Z) = d_i \cdot \left(\mathrm{N}_i^o(Z) - 1 \right) \tag{3.4}$$

and

$$l_i^s = d_i \cdot \left(n_i^s - 1 \right) \tag{3.5}$$

where $\mathrm{N}_i^o(Z)$ and n_i^s are 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.

Equation 3.3 assumes that a constant end-effector speed will be employed to achieve uniform coverage of the surfaces (e.g. when spray painting a vehicle). However, in certain situations and applications, non-uniform coverage may in fact be required. For example, when grit-blasting a steel object where only certain areas of the object are heavily rusted, then the rusted areas will need a more intensive, or extended period of grit-blasting to achieve a uniform surface finish. This area-specific focus can be accomplished by reducing the end-effector speed for such target areas. Each target can thus be weighted based on the surface area it represents (e.g. the level of rust on the area). This weighting can then

be used for calculating $\mathrm{T}_i(Z)$. That is,

$$\mathrm{T}_i(Z) = \sum_{j=1}^{\mathrm{N}_i^o(Z)} \frac{d_i}{(w_{ij} \cdot v_i^d) + v_i^{min}} + \sum_{j=1}^{n_i^s} \frac{d_i}{(w_{ij} \cdot v_i^d) + v_i^{min}} \tag{3.6}$$

where $v_i^d = v_i^{max} - v_i^{min}$, and where v_i^{max} and v_i^{min} are the maximum and minimum end-effector speeds of the ith AIR, respectively, and w_{ij} is the weighting factor, $0 \leq w_{ij} \leq 1$, applied to the end-effector speed based on the area in which the target $\boldsymbol{o}_{ij}$ is located.

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. Thus, as an alternative to Eq. (3.2), the equation below can be used which purely focuses on minimizing the makespan:

$$\min_Z \mathrm{F}_1(Z) = \max\big\{\mathrm{T}_1(Z), \mathrm{T}_2(Z), \ldots, \mathrm{T}_n(Z)\big\}. \tag{3.7}$$

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 affected 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:

$$\min_Z \mathrm{F}_2(Z) = \sum_{i=1}^{n} \sum_{j=1}^{\mathrm{N}_i^o(Z)} \left\| \boldsymbol{c}_i^s - \boldsymbol{o}_{ij} \right\|. \tag{3.8}$$

If there are no specific areas for an AIR, then the AIR's base location can be chosen instead of $\boldsymbol{c}_i^s$.

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 for this AIR the sum of distances of the target from the allocated areas to the specific areas can be large. Thus, a compromise solution may be necessary.

Objective 3 - Minimal 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 $k(k = 1, \ldots, n^J)$ 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

$$\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f) = \max_k \left| \frac{\mathcal{T}_{ik}(\boldsymbol{q}_{ij}^f)}{\tau_{ik}^c} \right| \tag{3.9}$$

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 torque 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. Appendix B provides a detailed explanation of how

torque is calculated. Since the motion of the AIR during task execution is slow in such applications, other factors such as angular, centripetal and Coriolis accelerations [129] 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.

$$\min_{Z} \mathrm{F}_3(Z) = \sum_{i=1}^{n} \sum_{j=1}^{\mathrm{N}_i^o(Z)} \mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f). \tag{3.10}$$

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 minimal 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.

Objective 4 - Maximal Manipulability Measure: Manipulability measure [130] 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 [131]. Manipulability measure of an AIR pose $\boldsymbol{q}_{ij}^f$ can be calculated as

$$\mathrm{W}(\boldsymbol{q}_{ij}^f) = \sqrt{\det\left(\mathbf{J}(\boldsymbol{q}_{ij}^f)\mathbf{J}^{\mathsf{T}}(\boldsymbol{q}_{ij}^f)\right)} \tag{3.11}$$

where $\mathbf{J}(\boldsymbol{q}_{ij}^f)$ 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.

$$\max_{Z} \mathrm{F}_4(Z) = \sum_{i=1}^{n} \sum_{j=1}^{\mathrm{N}_i^o(Z)} \mathrm{W}(\boldsymbol{q}_{ij}^f). \tag{3.12}$$

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 other AIR may already be covering a larger area. Additionally, the targets that are close to the specific 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.

3.3 Case Studies and Results

Eight case studies are presented in this section to validate the approach using different conditions and scenarios. The first case study is to validate the APA approach using 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 [62]. Case Study 3 shows the benefit of incorporating manipulator-related objectives such as torque minimization. Case Studies 4 and 5 validate the approach for a complex scenario where multiple objects are separated from each other (unconnected) and provide a discussion on the Pareto front. Case Study 5 also demonstrates a method for fixing missing sections when more than two AIRs are deployed. In Case Study 6, another scenario using 4 AIRs is presented to show that missing sections can be fixed using the procedure outlined in Case Study 5. Case studies 7 and 8 use data from a real-world application and real objects to validate the approach for an object with a complex geometric shape and a concave object, respectively.

Specifications of the AIRs used in the case studies are provided in Appendix A. 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. (3.3). Thus, the completion time does not include the preliminary computation times of the case study, e.g. the computation time associated with performing the optimization. The computation time associated with the optimization process of APA is included within each case study. A table is added in the next section (Section 3.4 - "Discussion") which summarizes the completion times of the AIRs and the computation times for the case studies.

Many optimization algorithms can be used to solve the optimization problem. Comparative studies between optimization algorithms and parameters are outside the scope of this thesis. An example of multi-objective optimization is Non-dominated Sorting Genetic Algorithm II (NSGA-II) [132], 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 optimization problem. In the following sub-section, 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. The optimization was repeated 10 times to check convergence and consistency of the results for the relevant case studies. Multi-threading is enabled for the optimization to use all four cores of the CPU. All other parameters are set as default.

3.3.1 Procedure for Calculating the Objective Functions

Function 3.2 is shown to provide a brief insight into the procedure for calculating the objective functions within the optimization algorithm. Although NSGA-II is used as an example optimization tool for solving the optimization problem, Function 3.2 can also 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 the n AIRs, and the targets $O^s = \{O_1^s, O_2^s, \ldots, O_n^s\}$ that represent the specific areas of the n AIRs. The function starts with looping 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

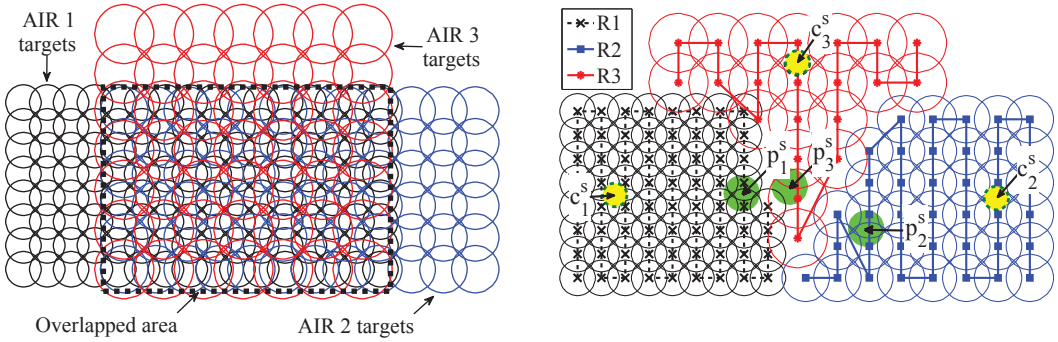
Function 3.2 Calculating Objective Functions.

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** $\|\boldsymbol{o}_{ij} - \boldsymbol{p}_i^s\| < \|\boldsymbol{o}_{ij} - \boldsymbol{p}_k^s\|$ **then**
5: $O_i^{al} \leftarrow O_i^{al} \frown \boldsymbol{o}_{ij}$
6: $\mathcal{T}_i^{al} \leftarrow \mathcal{T}_i^{al} \frown \mathfrak{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$
7: $W_i^{al} \leftarrow W_i^{al} \frown \mathrm{W}(\boldsymbol{q}_{ij}^f)$
8: **else**
9: $O_i^{rej} \leftarrow O_i^{rej} \frown \boldsymbol{o}_{ij}$
10: $\mathcal{T}_i^{rej} \leftarrow \mathcal{T}_i^{rej} \frown \mathfrak{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$
11: $W_i^{rej} \leftarrow W_i^{rej} \frown \mathrm{W}(\boldsymbol{q}_{ij}^f)$
12: **end if**
13: **end for**
14: **end for**
15: **if** $n > 2$ **then**
16: $Data \leftarrow [O^{al}, \mathcal{T}^{al}, W^{al}, O^{rej}, \mathcal{T}^{rej}, W^{rej}]$
17: $[O^{al}, \mathcal{T}^{al}, W^{al}] \leftarrow$ FixMissingSections($Data$)
18: **end if**
19: $\mathrm{F}_1 \leftarrow$ TimeVariance(O^{al})
20: $\mathrm{F}_2 \leftarrow$ Distance2Centroid(O^s, O^{al})
21: $\mathrm{F}_3 \leftarrow$ Sum($\mathcal{T}^{al}$)
22: $\mathrm{F}_4 \leftarrow -$Sum($W^{al}$)
23: **return** $[\mathrm{F}_1, \mathrm{F}_2, \mathrm{F}_3, \mathrm{F}_4]$
24: **end function**

inside the corresponding ith Voronoi cell, then the target $\boldsymbol{o}_{ij}$, the maximum torque ratio $\mathfrak{T}^{Rmax}(\boldsymbol{q}_{ij}^{f})$ experienced by a joint of the ith AIR at pose $\boldsymbol{q}_{ij}^{f}$ created to reach $\boldsymbol{o}_{ij}$, and the manipulability measure $\mathrm{W}(\boldsymbol{q}_{ij}^{f})$ 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 superscripts al and rej are used to symbolize allocated and rejected targets, respectively. $\mathfrak{T}^{Rmax}(\boldsymbol{q}_{ij}^{f})$ and $\mathrm{W}(\boldsymbol{q}_{ij}^{f})$ are calculated based on Eq. (3.9) and Eq. (3.11), respectively. Lines 15 to 18 are designed to deal with a special condition where Voronoi partitioning may cause incomplete coverage of the overlapped areas when more than two AIRs are deployed, if the overlapped areas between the AIRs are 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 Section 3.3.6. 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 calculated using the equations outlined in Section 3.2.2.2. In line 19, TimeVariance is a function representing Eq. (3.2). In line 20, Distance2Centroid is a function representing Eq. (3.8). 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 Eq. (3.10) and Eq. (3.12), respectively.

3.3.2 Case Study 1: Three AIRs with Different Capabilities to Grit-blast a Flat Plate

This case study demonstrates the effectiveness of the APA approach for conditions where AIRs' capabilities are different. Assume that the overlapped and specific areas of each AIR are as shown in Fig. 3.6a and are calculated based on the base locations and workspace size of the AIRs. The AIRs have different capabilities, meaning that they have different speed and tool size; therefore, the size of the targets for each AIR is different. 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 3.1.



(a) Overlapped and specific areas of each AIR. **(b)** Result and final path of each AIR.

Fig. 3.6: Three AIRs with different capabilities to operate on a flat surface.

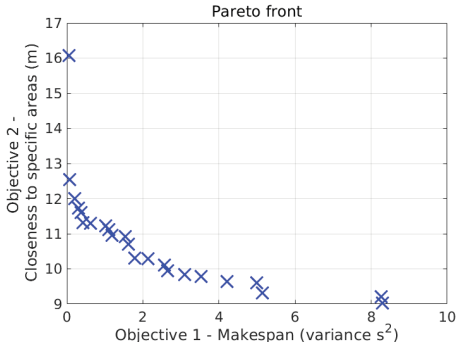


Fig. 3.7: Pareto front from one of the optimization runs.

Objectives 3 and 4 are discarded in this case study. A solution from the Pareto front (shown in Fig. 3.7) that results in the minimal overall completion time (makespan) of the task is chosen. The completion time of AIRs 1 to 3 is 13.5 s, 13.6 s and 13.4 s, respectively. It can be seen from Fig. 3.6b that the allocated areas are close to their corresponding AIRs due to incorporating Objective 2 in the optimization model. 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.

Table 3.1: Parameters related to 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

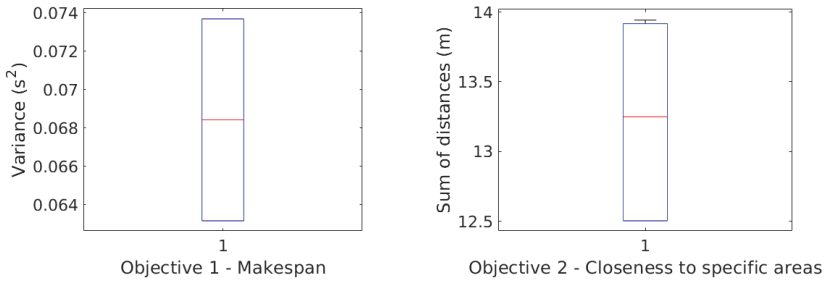


Fig. 3.8: Boxplots of the two objectives for the 10 optimization runs.

Convergence and consistency: To check convergence and consistency of the results, the optimization was repeated 10 times. For each run, a solution from the Pareto front that results in the minimal makespan is chosen. The solution shown in Fig. 3.6b is based on one of the 10 optimization runs. Figure 3.8 shows the boxplots of the two objectives for the 10 runs. From the 10 runs, the average of the overall completion times of the task is 13.8 s (0.3 s less than the optimal). The computation time for the optimization is less than 10 s on average.

Search space and scalability: The search space is continuous; however, bounded by a bounding rectangle (or a bounding box) that occupies the overlapped areas. Thus, the seed points of the Voronoi cells are constrained to be within this bounding rectangle/box. Note that from Section 3.2.2.1, the design variables are the position coordinates of the seed points $\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \ldots, \boldsymbol{p}_n^s$. Thus, for n AIRs, there are $3n$ design variables. For a planar surface, one of the coordinates of the seed points will be a constant.

The search space can be discretized to improve computational efficiency. Suppose that the search space is uniformly discretized into G grids. The seed points can then be constrained to be only at the center of the grids. Thus, there are $\frac{G!}{(G-n)!}$ permutations for the search space where n is the number of AIRs. As an example, let the overlapped area of the planar surface presented in this case study be discretized based on the smallest target size (i.e. AIR 1) which gives 96 grids (12 x 8). Thus, there are 857280 permutations. Note that this is a large search space for a planar surface with small overlapped areas and coarse discretization. Three-dimensional space, large overlapped areas, or large number of AIRs

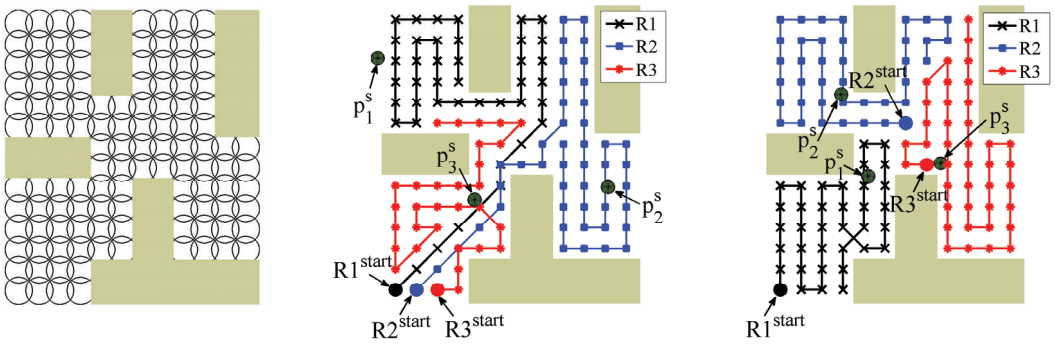
can significantly increase the size of the search space. Hence, exhaustive search through the search space is not practical.

3.3.3 Case Study 2: Comparing the APA Approach to the Pattern-Based GA Approach

The 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 [62], which performs partitioning and CPP concurrently. In pattern-based GA, eight patterns are designed to "provide disciplined, reasonable rectilinear moves"[62] for each robot. The approach has been validated using flat rectilinear surfaces, and it has been shown to achieve significant improvements over Hierarchical Oriented GA (HOGA).

The 8.4 m $\times$ 7.2 m environment shown in Fig. 3.9a 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 Figs. 3.9b and 3.9c for scenarios 1 and 2, respectively. Kapanoglu et al. [62] 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. [62] consider the completion time of a robot to contain "the linear moving times between 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



(a) The test environment used for comparisons.

(b) Scenario 1.

(c) Scenario 2.

Fig. 3.9: 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 APA approach are shown for two scenarios where in each scenario different initial start points are considered for the robots.

(90°)]*t_{ra}}. Same as in Kapanoglu et al. [62], robots' velocity 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 APA approach, the paths shown in Figs. 3.9b and 3.9c are obtained where the overall completion time is 145.9 s ($\sim 1.5\%$ improvement) and 134.5 s ($\sim 2\%$ improvement) for scenarios 1 and 2, respectively. Although the improvement may seem small, note that the results presented by [62] using the pattern-based GA are already near-optimal. The improvements made by the APA approach is due to the non-rectilinear moves (diagonal moves) of the robots, as can be seen in Figs. 3.9b and 3.9c. 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 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. 3.9b).

3.3.4 Case Study 3: Two AIRs to Grit-blast a Flat Plate in the Presence of an Obstacle

The purpose of this case study is to demonstrate how the APA approach can help with minimizing the torque experienced by the joints of the AIRs. Objective 4 (manipulability measure) is not considered in this case study. Fig. 3.10 shows a flat surface which is 2 m in length and 1.2 m in width. It is positioned 1.5 m above the base of the AIRs. Both AIRs are equipped with the same size nozzle and hence, the diameter of the targets representing the surface is the same. The AIRs operate with the end-effector speed, v, set to 0.1 m/s relative to the surface. The distance, d, between the centers of two adjacent targets along a path is 0.0563 m.

The case study was performed twice, i.e. with and without incorporating the objective of minimizing the joints' torque (Objective 3). Figure 3.11a shows a solution for the simulation where torque minimization is not considered, whereas the solution that is shown in Fig. 3.11b is based on a simulation that considers torque minimization. The solutions are chosen from the Pareto front such that Objective 1 (minimal makespan) is near-optimal. For both simulations, solutions that have the same completion times are selected. The completion times of AIR 1 and AIR 2 is 203 s and 201 s, respectively. It is clear that the difference in the completion time of the two AIRs is insignificant; hence, Objective 1 is near-optimal. Optimization time for this simulation is less than 25 s.

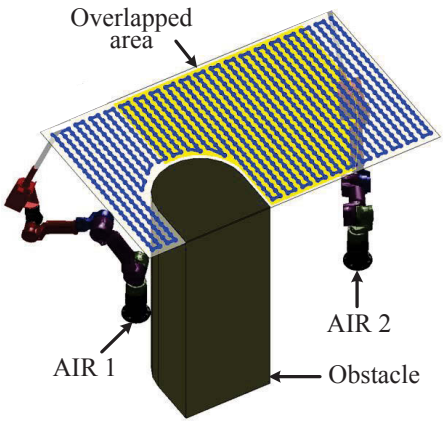
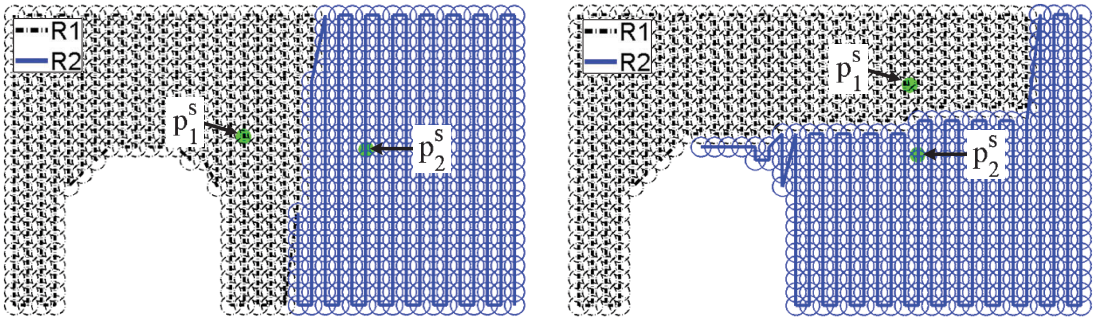


Fig. 3.10: Two AIRs operating on a flat plate.

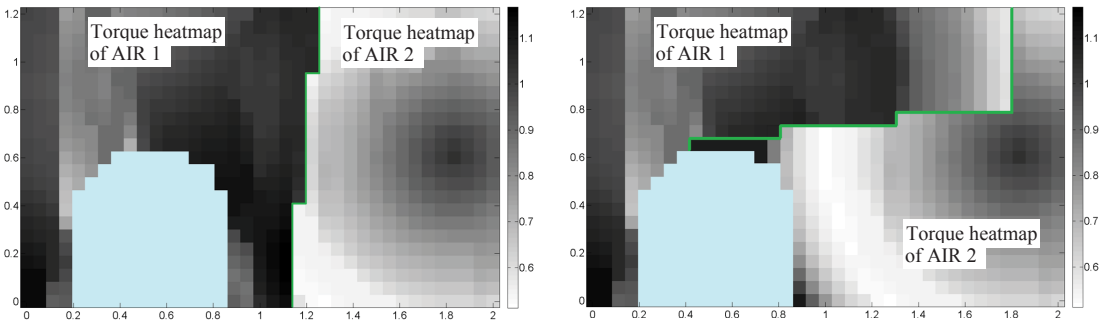


(a) A solution from the Pareto front of the first simulation (Toruqe minimization not considered).

(b) A solution from the Pareto front of the second simulation (Toruqe minimization considered).

Fig. 3.11: Results from two simulations.

Figure 3.12 illustrates the torque heatmaps of the solutions shown in Fig. 3.11. The torque due to AIRs' poses is used to generate the torque heatmaps. For each AIR pose corresponding to a target, only the maximum torque ratio, $\mathcal{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$, which is the largest value from all the torque ratios of the joints (actuators) of the AIR is taken into account when producing the heatmaps. The darker areas in Fig. 3.12 illustrate large torque regions. It is clear that Fig. 3.12b shows better results in terms of torque since the selection of large torque regions from the overlapped areas is significantly reduced, particularly for AIR 1. Hence, incorporating the torque minimization objective in the mathematical modeling can be beneficial.



(a) Torque heatmap of simulation 1.

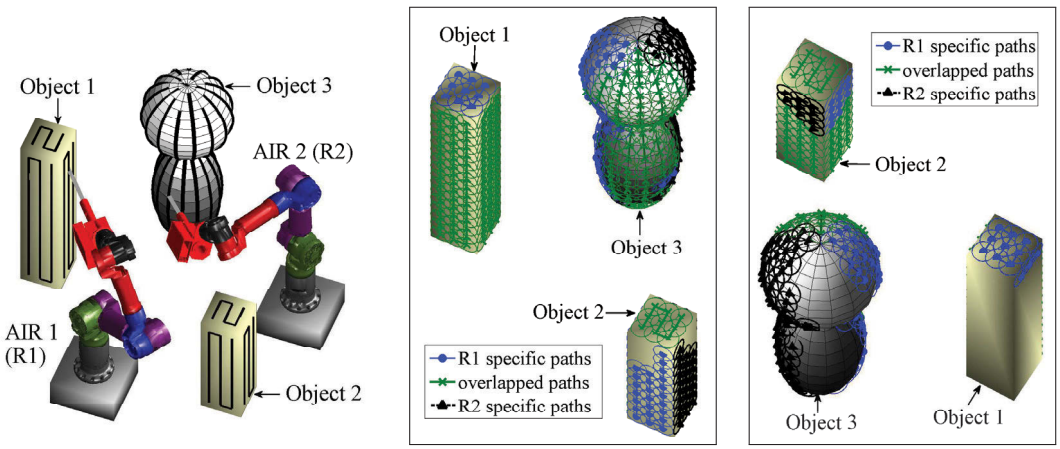
(b) Torque heatmap of simulation 2.

Fig. 3.12: Torque heatmaps of the two simulations.

3.3.5 Case Study 4: Two AIRs Spray Painting Three Separated (Unconnected) 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. Since multi-objective optimization is used, then a solution from the Pareto front needs to be selected. 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.

Consider that the three objects shown in Fig. 3.13a are to be spray painted by the two deployed AIRs. The targets and the paths created on the overlapped and specific areas of the two AIRs are shown in Figs. 3.13b and 3.13c. These paths will be modified based



(a) Two AIRs spray painting three seperated (unconnected) objects.

(b) Overlapped and specific areas (view angle 1).

(c) Overlapped and specific areas (view angle 2).

Fig. 3.13: Overlapped and specific areas of two AIRs which will be used to spray paint three objects.

on the chosen solution from the Pareto front. In Figs. 3.13b and 3.13c, the areas that 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 at their current base placement.

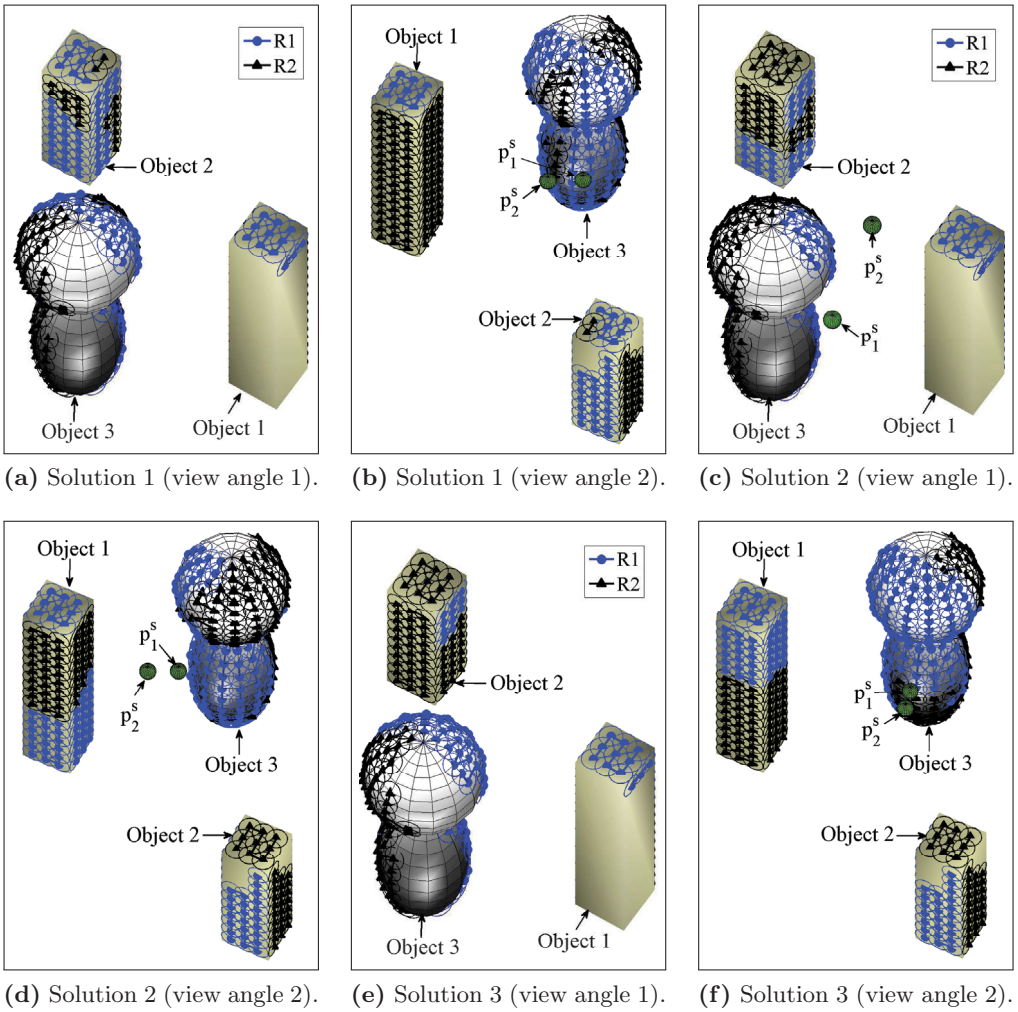
The two AIRs are assumed to be identical and therefore, both AIRs have the following properties: the radius of the targets (r^o in meters) is 0.04, the distance between two adjacent targets along a path (d in meters) is 0.0563, and the end-effector speed (v in meters per second) is 0.1.

After running the optimization, a small set of solutions from the Pareto front that are best in terms of the overall completion time (Objective 1) are first chosen. Three of these solutions are shown in Table 3.2 and Fig. 3.14 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 3.2, 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 solutions 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 Figs. 3.14e and 3.14f corresponding to solution 3, with Figs. 3.14a to 3.14d corresponding to solutions 1 and 2, it is clear that the targets are better grouped with each other (in Figs. 3.14e and 3.14f) 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 help the AIRs with better motion during the task execution.

Table 3.2: Three solutions from the Pareto front.

Soln. No.	Obj. 1: makespan (s^2)	Obj. 2: proximity (m)	Obj. 3: torque (N.m)	Obj. 4: manipulability
1	0	142.2	120.1	-16.4
2	0	150.6	124.5	-16.5
3	12.1	138.6	123.5	-15.1



(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).

Fig. 3.14: AIRs' final paths created on the three objects based on three solutions chosen from the Pareto front.

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 3.3.

Table 3.3: Completion time of the AIRs in seconds.

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

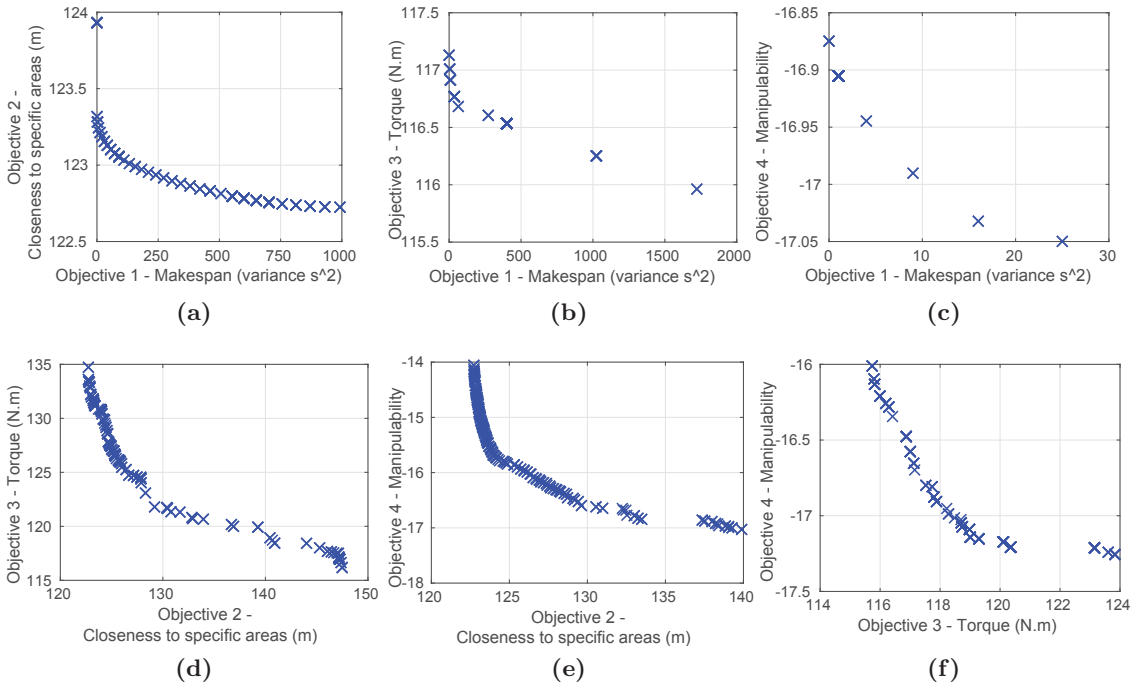


Fig. 3.15: Trade-off for all combinations of the objectives.

Trade-off between objectives: To give an insight into the trade-off between the objectives, the optimization is repeated for all pairs of objective functions and the Pareto front is plotted (Fig. 3.15). The plots confirm that all objectives are in conflict with each other.

Convergence and consistency: The optimization was repeated 10 times to check convergence and consistency of the results. For each run, a solution from the Pareto front that results in the minimal makespan is chosen. The average of the 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 of the overall completion times of the task is 120 s (optimal). Figure 3.8 shows the boxplots of Objectives 3 to 4 for the 10 runs. The computation time for the optimization is less than 15 s on average.

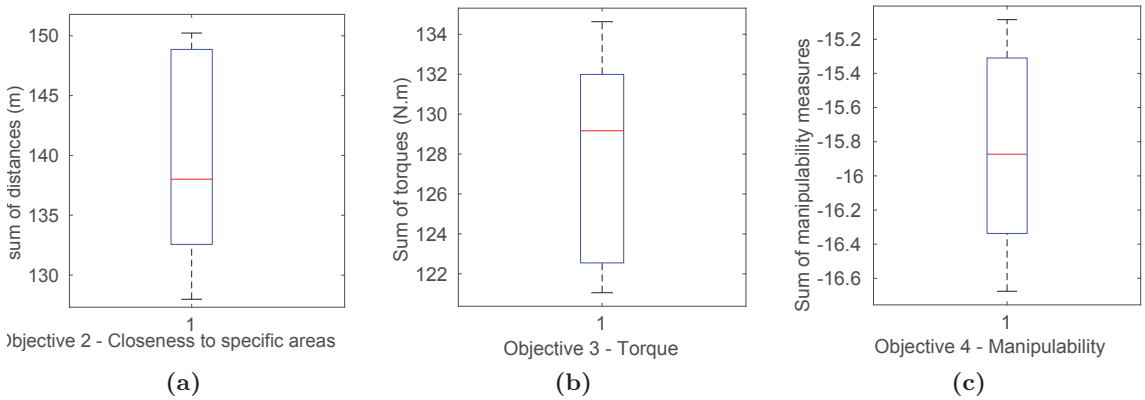


Fig. 3.16: Boxplots of Objectives 2 to 4 for the 10 optimization runs.

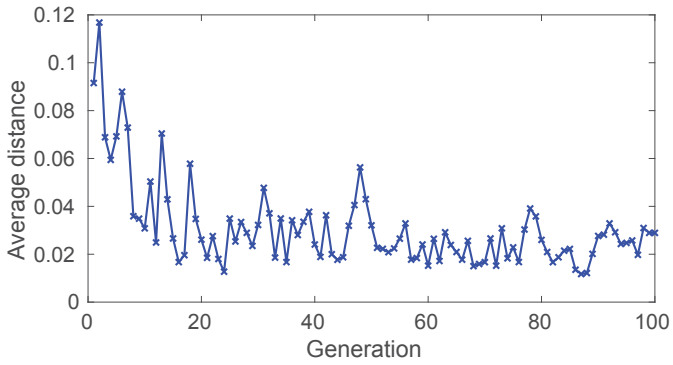


Fig. 3.17: Average of distances of individuals at each generation.

Figure 3.17 is created from an optimization run. The figure shows the average of distances of individuals at each generation using the field 'AverageDistance' of Matlab 'gamultiobj'. At each generation, the average of distances of each member of the population to the nearest neighboring member is calculated and saved to the field 'AverageDistance'. The average of distances tends to favorably decrease as the generation number increases and eventually converges to approximately 0.02. Note that NSGA-II tries to diversify the solutions at each generation in the hope of exploring the unexplored regions of the search space, hence the reason for the spikes.

3.3.6 Case Study 5: 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 4 is used; however, an additional AIR that is identical to the other two AIRs is now introduced in the environment as shown in Fig. 3.18. 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: (i) more than two AIRs are deployed to carry out the task, and (ii) 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 Figs. 3.19a and 3.19b. 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 3.2 and then the function will check for and fix missing sections by running Function 3.3.

Function 3.3 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

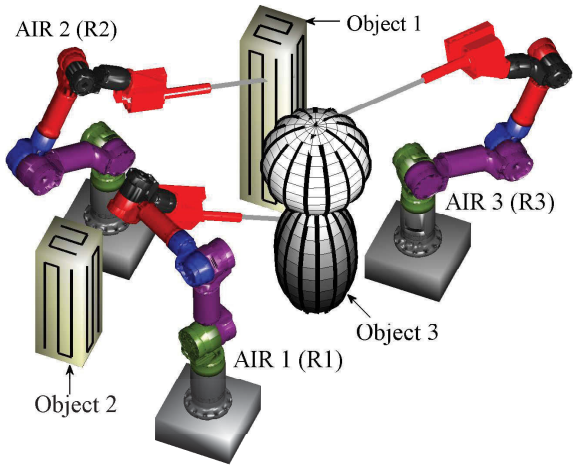
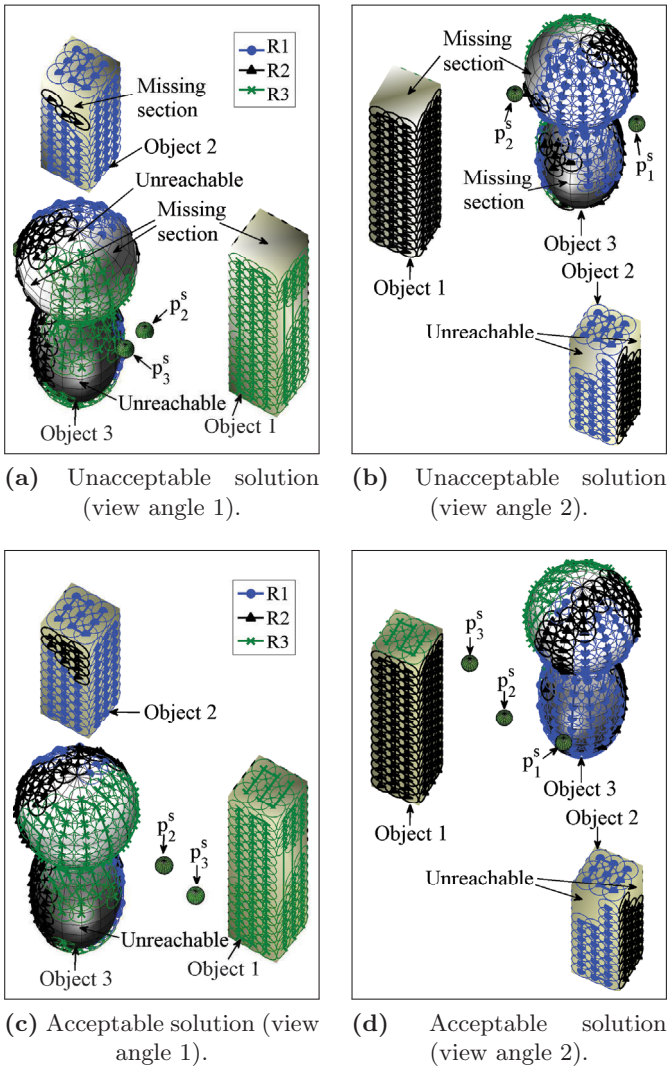


Fig. 3.18: Three AIRs are used to operate on three different objects which are separated from each other.

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}^{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: (i) 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



(a) Unacceptable solution (view angle 1).

(b) Unacceptable solution (view angle 2).

(c) Acceptable solution (view angle 1).

(d) Acceptable solution (view angle 2).

Fig. 3.19: 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.

target that can cover most of 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 distance is less than r_{ij}^{o}, i.e. if $\boldsymbol{o}_{ij}$ overlaps with $\boldsymbol{o}_{i'j''}$), (ii) calculating the distances between all targets $\boldsymbol{o}_k \in Nearest$ and their corresponding Voronoi seed point $\boldsymbol{p}_k^s$, and obtaining the minimum distance d^{min} from the calculated distances, and (iii) if d^{min} is greater than the distance between the target $\boldsymbol{o}_{ij}$ and its corresponding Voronoi seed point $\boldsymbol{p}_i^s$, then the area a_{ij}^{rej} is allocated to the target $\boldsymbol{o}_{ij}$. Lines 8 to 10 add the target $\boldsymbol{o}_{ij}$, the maximum torque ratio $\mathfrak{T}^{Rmax}(\boldsymbol{q}_{ij}^f)$, and the manipulability measure $\mathrm{W}(\boldsymbol{q}_{ij}^f)$, to the sets O_i^{al}, $\mathcal{T}_i^{al}$ and W_i^{al}, respectively. Note that in order to make this function time efficient, the use of bounding volumes or voxels [133] and grouping the targets in a hierarchical data structures such as Quadtrees or Octrees [134] is needed for reducing the number of distance queries between the targets (e.g. in lines 4 and 5) and acquiring data. Function 3.3 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.

Function 3.3 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
```

Continuing with the example scenario, the additional procedure for fixing missing sections helped with obtaining complete coverage as shown in Figs. 3.19c and 3.19d which are based on a solution chosen from the Pareto front. In Figs. 3.19c and 3.19d, the areas that 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 (optimal).

The optimization was repeated 10 times to check convergence and consistency of the results. The above solution is based on the output (Pareto front) of one of the 10 optimization runs. The average of the 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 of the overall completion times of the task is 109 s (optimal). The computation time for the optimization is less than 2 minutes on average.

3.3.7 Case Study 6: Four AIRs with Different Overlapped Areas

In this case study, another scenario using 4 AIRs is presented to show that missing sections can be fixed using the procedure outlined in Case Study 5 (Section 3.3.6). The AIRs have different capabilities and hence, the size of the targets associated with each AIR are different. The AIRs operate with the end-effector speed, v, set to 0.1 m/s relative to the surface. The distance, d, between two adjacent targets along a path associated with AIRs 1 to 4 is 0.0603 m, 0.0469 m, 0.0335 m, and 0.0737 m, respectively. The reachable and overlapped areas of the AIRs are shown in Fig. 3.20. As explained previously, when more than two AIRs are deployed and if the overlapped areas are different amongst the

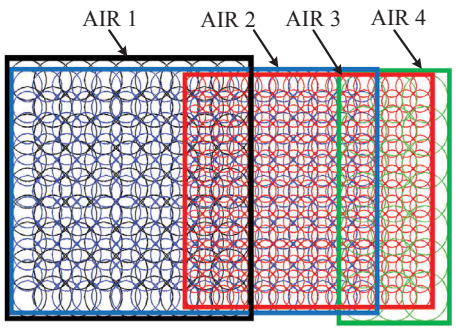
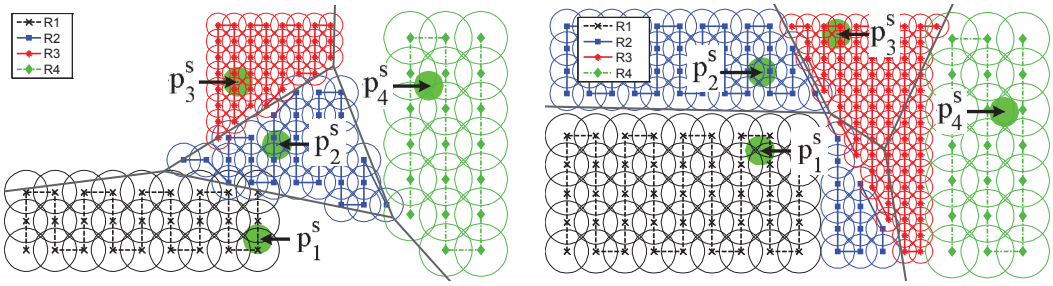


Fig. 3.20: Reachable and overlapped areas of four AIRs with different capabilities.



(a) Unacceptable solution due to unreachable sections in the allocated cells.

(b) Acceptable solution through further allocation of missing sections.

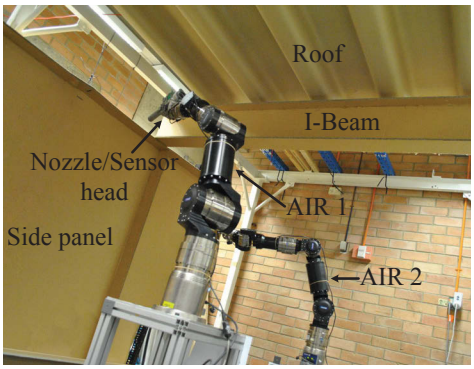
Fig. 3.21: Missing sections of four AIRs are fixed through further allocation.

AIRs, then missing sections can be caused. To help with demonstrating the problem, only Objective 1 (minimal makespan) is used.

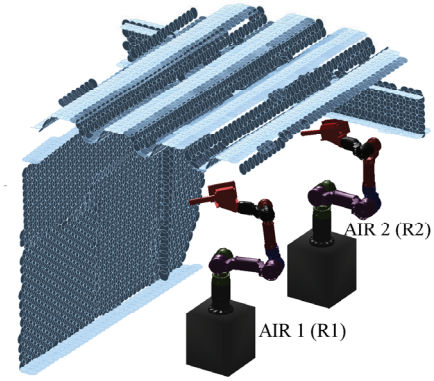
Missing sections occur due to the fact that some sections of the allocated cells can't be covered (unreachable) by their corresponding AIRs. This condition is shown in Fig. 3.21a which is based on a solution where the additional procedure for fixing the missing sections was not taken into account in the optimization. Clearly, this result is unacceptable due to incomplete coverage. After taking into account the procedure explained in Section 3.3.6, an acceptable solution is obtained as shown in Fig. 3.21b. Completion time of the AIRs for this solution are 26.5 s, 25.8 s, 26.5 s, and 14.7 s for AIRs 1 to 4, respectively. Since AIR 4 can reach a smaller area of the surface, it is preferred that it covers all of its reachable areas in order to minimize the makespan. As expected, simulation result confirms that AIR 4 is allocated all of its reachable areas. The computation time for this case study is less than 6 minutes on average.

3.3.8 Case Study 7: Two AIRs Used to Grit-blast a Small Area of a Steel Bridge

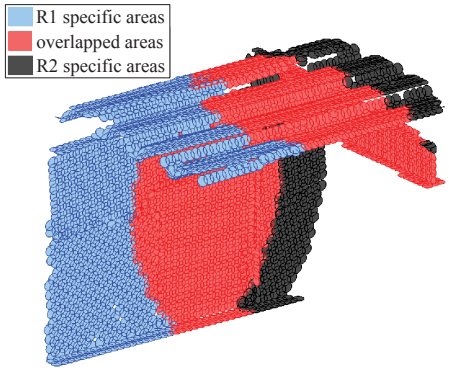
The APA approach is tested by using real data generated from the grit-blasting application where rust and other debris are removed from objects' surfaces. In this application, the grit exits the nozzle with a high pressure and helps with cleaning the surfaces. The deployed AIRs are identical and are equipped with the same end-effector nozzle. The AIRs operate with the end-effector speed, v, set to 0.1 m/s relative to the surface. The radius, r^o, of the



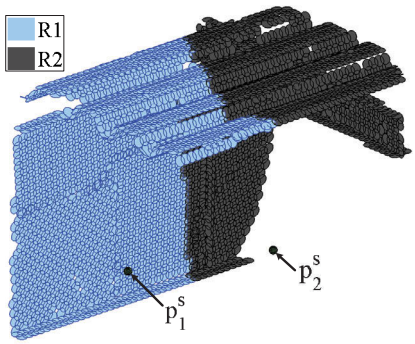
(a) Two identical AIRs to be tested in a cardboard test rig replicating part of a steel bridge.



(b) Target representation of the sufaces 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.

Fig. 3.22: 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.

targets associated with all AIRs is set to 0.04 m, and the distance, d, 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. 3.22a, where two 6 DOF AIRs are being tested prior to being deployed in the real environment. 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. 3.22b.

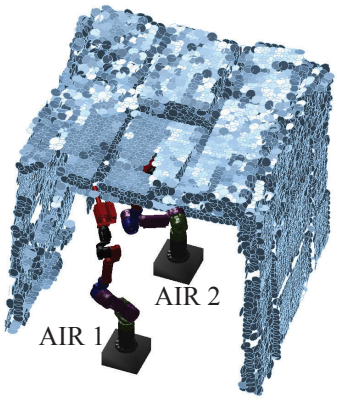
Figure 3.22c shows the specific and overlapped areas associated with the two AIRs. Figure 3.22d is based on a solution chosen from the Pareto front. This solution is chosen by considering that the overall completion time (Objective 1) has the highest priority. The solution is optimal in terms of overall 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 to check convergence and consistency of the results.. The above solution is based on the output (Pareto front) of one of the 10 optimization runs. The average of the 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 of the overall completion times of the task is 969 s (optimal). The computation time for the optimization is less than 1 minute on average.

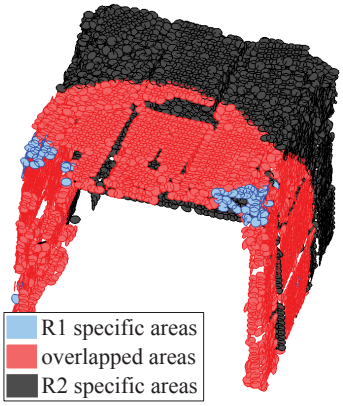
3.3.9 Case Study 8: Two AIRs Used to Grit-blast a Boxlike Steel Structure

The APA approach is tested again using a real concave boxlike steel structure. As shown in Fig. 3.23a, 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 are shown in Fig. 3.23b. The specific and overlapped areas, shown in Figs. 3.23c and 3.23d, 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 Figs. 3.23c and 3.23d 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 study, introducing a second AIR to the environment is mainly to reduce the overall completion time rather than achieving a greater coverage.

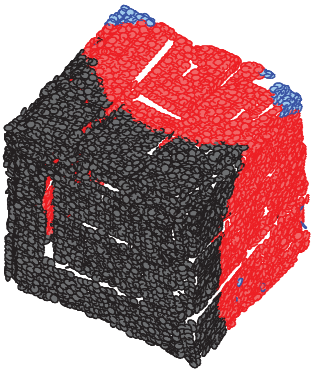
(a) An AIR performing scanning and exploration of the environment.



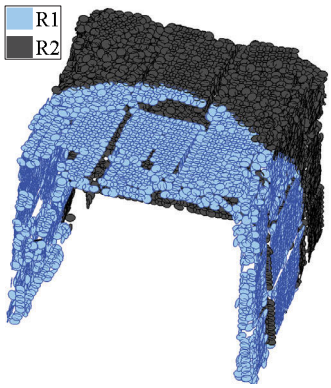
(b) Target representation and the location of the two AIRs.



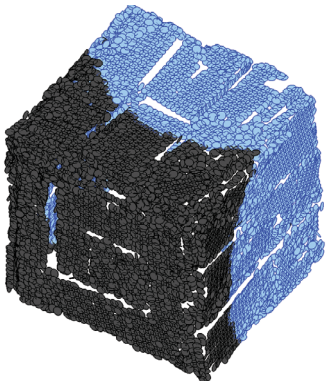
(c) Overlapped and specific areas (view angle 1).



(d) Overlapped and specific areas (view angle 2).



(e) Final solution (view angle 1).



(f) Final solution (view angle 2).

Fig. 3.23: Overlapped and specific areas as well as the final solution associated with the two AIRs which will be used to grit-blast a concave boxlike steel structure.

A solution from the Pareto front is shown in Figs. 3.23e and 3.23f. 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 to check convergence and consistency of the results. The above solution is based on the output (Pareto front) of one of the 10 optimization runs. From the 10 optimization runs, the average of the overall completion times of the task is 1829 s (9 s less than the optimal). The computation time for the optimization is less than 2 minutes on average.

A video is provided ($https://youtu.be/1TjrcXKpy4$)[2] for this case study to recap the steps for obtaining the results. The video also includes the environment exploration phase.

3.4 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 3.4 summarizes the makespan of the case studies. Note that the area to be covered by the AIRs is fixed. Thus, given the number of AIRs and the number of targets representing the surface, the optimal makespan can be calculated for comparison with the actual makespan of each case study. Table 3.4 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. However, potential improvements in the computation time can be investigated as 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 further studied.

[2] A video for Case Study 8 can be viewed at $https://youtu.be/1TjrcXKpy4$.

Table 3.4: 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	203	1 / 0.5	25
4	120	Optimal	15
5	109	Optimal	120
6	26.5	0.2 / 0.8	360
7	969	Optimal	60
8	1829	9 / 0.5	120

3.5 Conclusions

The APA approach presented in this chapter was aimed at simultaneously partitioning and allocating the overlapped areas amongst the AIRs. In doing so, the capabilities of the AIRs such as speed and tool's coverage size were considered. Four objectives were taken into account, and they are as follow: (i) minimal overall completion time of the task, (ii) minimal closeness of the allocated areas to the corresponding AIR, (iii) minimal torque experienced by the AIRs' joints, and (iv) 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. Many case studies 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, it will be interesting to improve the computational efficiency of the approach by investigating aspects such as discretizing the search space and comparing different optimization algorithms.

Chapter 4

An Optimization-Based Method to Multi-AIR Base Placement

Appropriate base placements of AIRs relative to the objects need to be determined. The problem of finding appropriate base placements for AIRs is complicated when the object under consideration is large and/or has a complex geometric shape, and thus the AIRs need to operate from a number of base locations in order to achieve complete coverage of the entire object. To address this problem, a method for Optimization of Multiple Base Placements (OMBP) for each AIR is presented in this chapter. The method aims to optimize AIRs' base placements by taking into account task-specific objectives such as makespan, fair workload division amongst the AIRs, and coverage percentage; and manipulator-related objectives such as torque and manipulability measure. In addition, the constraints of AIRs maintaining a safe distance between each other and relative to the objects is taken into account. Simulations and real-world experiments are carried out to test the effectiveness of the method and to verify that the results are accurate and reliable.

There is a number of research works in the literature for finding an appropriate base placement for a single robot in different environments, such as the manufacturing environments [106, 107] and underwater environments [108, 109]. Many of the existing methods utilize optimization techniques. The problem of finding an optimal base placement for an industrial robot can be computationally expensive due to the size of the search space. Hence, researchers often simplify the problem by considering only a limited number of critical

discrete end-effector positions [110]. However, for complete coverage tasks where a large number of points need to be reached, considering optimizing for only a few critical end-effector positions is not practical. The problem becomes increasingly complicated when (i) multiple robots are involved in carrying out the intended tasks, and (ii) each robot must find multiple base placements such that the robot team can collectively complete the overall task. For multiple AIRs, the team's objectives and constraints need to be taken into account.

The main results of this chapter were previously included in the following papers: Hassan et al. [24], Hassan et al. [25] and Hassan and Liu [23].

The remainder of the chapter is structured as follows. Section 4.1 provides a detailed description of the problem. Section 4.2 presents the methodology, consisting of three sub-sections: 4.2.1 provides an overview of the method, 4.2.2 details the mathematical model, and 4.2.3 presents an approach for solving the optimization problem where a multi-objective Genetic Algorithm is used. Simulations using three AIRs are performed in Section 4.3 to test the method. A real-world experiment is also conducted in Section 4.3 to validate the method. Section 4.4 provides a brief discussion on the results. Concluding comments are stated in Section 4.5.

4.1 Problem Definition

Fig. 4.1 shows an example application where two AIRs are deployed to clean an I-beam's surfaces through grit-blasting, prior to spray painting the surfaces. At their current base placements, the AIRs can collectively cover all surfaces of the I-beam by following the boustrophedon paths, and thus achieve the complete coverage goal. However, from a different set of base placements, it may be impossible to achieve complete coverage. Thus, optimizing the base placements for each AIR relative to the object and other AIRs is crucial.

The problem of AIR's base placement is further complicated when the object is larger than the AIR's workspace or has a complex geometric shape. For example, consider the environment shown in Fig. 4.2 where two AIRs are deployed to cover all surfaces of a

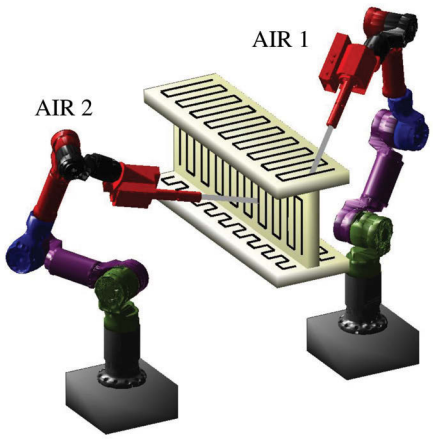
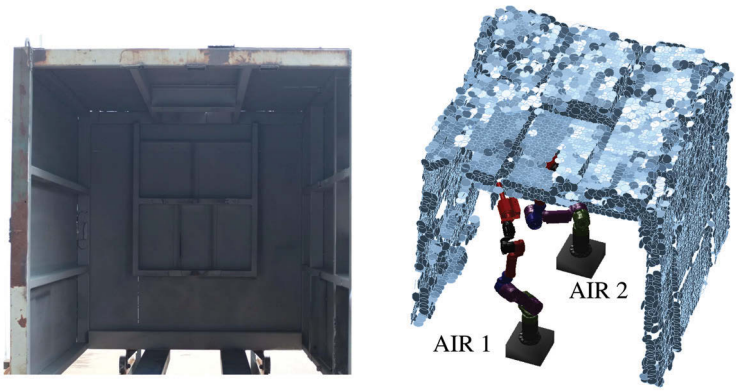


Fig. 4.1: The bases of two AIRs that are to grit-blast an I-beam are positioned appropriately relative to the I-beam and each other so as to jointly achieve complete coverage.

boxlike structure, including the internal and external surfaces. Each AIR needs to be repositioned several times so as to achieve complete coverage. Therefore, the problem is to find: (i) the minimal number, n^v of base placements for each AIR; (ii) the location of the n^v base placements for each AIR; and (iii) the visiting sequence of the n^v base placements.

The base placement problem must be solved while considering the following:

- Complete (or above threshold) surface coverage of the objects under consideration.
- Minimal overall completion time (or makespan).



(a) A boxlike structure to be operated on.

(b) Simulated scenario of the boxlike structure with two AIRs.

Fig. 4.2: Two AIRs to cover all internal and external surfaces of a boxlike structure.

- Fair workload division between the AIRs.
- Safe distance between the AIRs, and relative to the objects.
- Minimal torque experienced by the AIRs' joints.
- High manipulability so as to improve the AIR's ability to position and re-orientate its end-effector in different directions.

4.2 Methodology

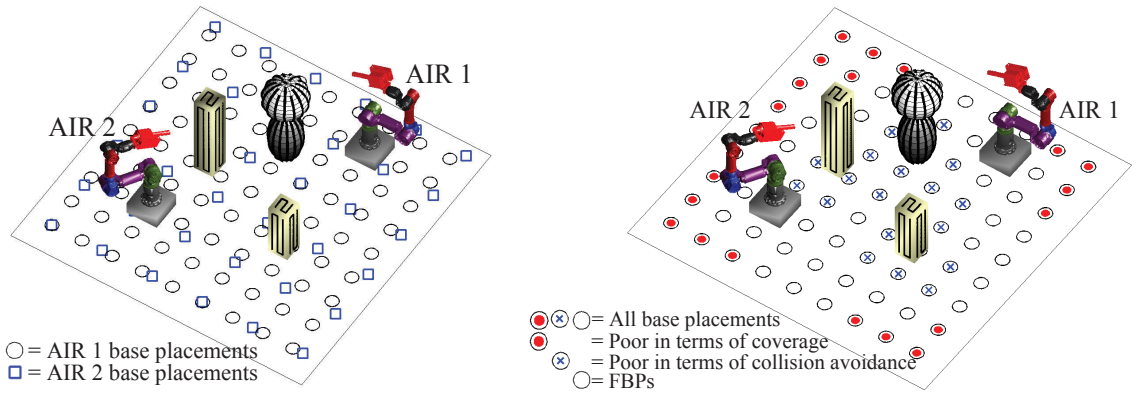
4.2.1 The OMBP Method

Algorithm 4.1 provides an overview of the OMBP method. The information obtained from exploring the environment is used in the OMBP method. As shown in line 1 of the algorithm, the OMBP method starts by discretizing the search space for two main reasons: (i) the object can be large or complex; thus multiple base placements for each AIR are needed, and (ii) complete coverage requires finding feasible AIR poses for a large number of points which is a computationally expensive process.

Fig. 4.3a shows two sets of candidate base placements, $B_i = \{\boldsymbol{b}_{i1}, \boldsymbol{b}_{i2}, \ldots, \boldsymbol{b}_{i(n_i^b)}\}$ for $i = 1, 2$ where i is the AIR's index and n_i^b is the total number of discrete base placements for the ith AIR. A base placement is defined as the x, y, z position with respect to a reference point. For each AIR, the number of candidate base placements and their density can be decided based on the application and the capacity of the AIR such as its workspace size.

Algorithm 4.1 OMBP.

1: For each AIR, create n_i^b discrete base placements around the target objects where i is the AIR index
2: Find favored base placements (FBPs)
3: Perform multi-objective optimization
4: Select the appropriate/best solution, Z^* from the Pareto front
5: **if** a threshold is not met **then**
6: Perform finer discretization around desirable FBPs
7: **goto** line 3
8: **end if**
9: **return** solution Z^* for execution



(a) Discrete candidate base placements of two AIRs.

(b) FBPs of AIR 1.

Fig. 4.3: Discrete candidate base placements of two AIRs with different capacity, and the FBPs of the first AIR.

For example, in Fig. 4.3a, the two AIRs have different capacity; thus the density of the candidate base placements can be different for each AIR. If the two AIRs were identical, then the candidate base placements can be the same for both AIRs. One challenge is to select a number of base placements from the set B_i for each AIR such that the team objectives are optimized. However, prior to performing the base placements optimization, simple preliminary filtering can prevent potentially poor performing base placements from becoming candidates, hence reducing the size of the search space. For example, Fig. 4.3b shows the candidate base placements of the first AIR. The base placements with a cross are deemed to be too close to the objects and are discarded to prevent the AIR from having a high likelihood of collision with the objects. The base placements indicated with filled red circles are also discarded because they are far away from the objects and consequently the coverage of the AIRs would be low. The rest of the base placements have reasonable to high coverage. These base placements are henceforth referred to as the Favored Base Placements (FBPs).

After determining the FBPs for each AIR (line 2 of Algorithm 4.1), the next step is to perform the multi-objective optimization (line 3). The aim is to select a subset of FBPs for each AIR and to determine the visiting sequence of the selected FBPs such that the team objectives are optimized, and constraints are satisfied.

The output of the multi-objective optimization is the Pareto optimal solutions that lie on

the Pareto front. Thus, a solution from the Pareto front needs to be selected (line 4). The Pareto front will be discussed in Section 4.2.3.

Coarse discretization for the base placements can be initially considered; however, if certain thresholds are not met then a finer discretization can be generated around the FBPs, and the optimization process is repeated (lines 5 to 8). For example, if the overall coverage of the object is below a predefined threshold, then finer discretization may produce better solutions. When producing finer discretization, an option is to use the best FBPs from the previous optimization run as part of the initial solution/s for the next optimization run. For example, if Genetic Algorithm (GA) is used, then the initial population can be made up of the best solutions from the previous run combined with the newly generated candidate base placements. Using the OMBP method, the location and the visiting sequence of the FBPs that each AIR needs to visit are determined.

4.2.2 Mathematical Model

4.2.2.1 Design Variables

Let $B_i^{FBP} = \{\boldsymbol{\beta}_{i1}, \boldsymbol{\beta}_{i2}, \ldots, \boldsymbol{\beta}_{i(n_i^F)}\} \subseteq B_i$ be the FBPs associated with the ith AIR, for $i = 1, 2, \ldots, n$. Note that for the ith AIR, $n_i^F \leq n_i^b$, meaning that the number of FBPs are less than or equal to the total number of discrete base placements. The design variables are $Z_{ik} \in \{0, 1, \ldots, n_i^F\}$ such that $Z_{ij} \neq Z_{ik} \iff Z_{ik} > 0$ where $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n_i^F$, $k = 1, 2, \ldots, n_i^F$, $k \neq j$. As an example, the design variable $Z_{ik} = 3$ means that the kth base placement of the ith AIR is the third FBP, i.e. the ith AIR is to visit $\boldsymbol{\beta}_{i(Z_{ik})} = \boldsymbol{\beta}_{i3}$ for its kth base placement. Hence, the ith AIR can visit up to a maximum of n_i^F FBPs where n_i^F is the total number of FBPs. If a design variable is given a value of zero, i.e. if $Z_{ik} = 0$, then one less FBP will be visited by the ith AIR and the AIR will move from the $(k-1)$th base placement $\boldsymbol{\beta}_{i(Z_{ik-1})}$ to the $(k+1)$th base placement $\boldsymbol{\beta}_{i(Z_{ik+1})}$. Let Z be a set containing all the design variables that have a value greater than 0, i.e. $Z = \{Z_{ik} | Z_{ik} > 0, \forall i, k : i = 1, \ldots, n, k = 1, \ldots, n_i^F\}$. The AIRs collectively visit n^v FBPs where n^v equals the number of design variables in Z. Similarly, n_i^v is defined as the number of FBPs to be visited by the ith AIR only, and can be determined based on the

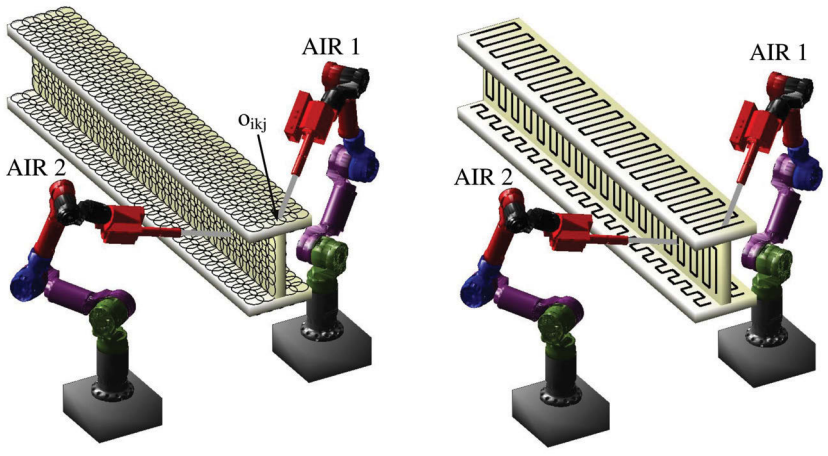
number of nonzero design variables that are associated with the ith AIR. Ultimately, the aim is to obtain values for n^v nonzero design variables such that the team objectives are optimized while constraints are satisfied.

It needs to be noted that since the number of FBPs to be visited by an AIR is initially unknown, the extreme, but the unlikely case would necessitate visiting all FBPs, and hence the number of design variables can be as large as the number of FBPs. However, an approximation of the number of FBPs to be visited by each AIR, i.e. n_i^v, can be made based on the size of the object as will be explained in Section 4.2.3. This approximation can significantly reduce the size of the search space.

4.2.2.2 Objective Functions

The main objectives that are relevant to the task of complete coverage and the performance of the AIRs are (i) maximal coverage, (ii) minimal makespan, (iii) maximal manipulability measure, and (iv) minimal joints' torques.

Objective 1 - Maximal Coverage: It is vital that the base placements selected by the AIRs result in maximum coverage of the surfaces. Figure 4.4 shows two AIRs that



(a) An I-beam represented using targets.

(b) Paths generated on the surfaces of the I-beam by joining targets' centroid.

Fig. 4.4: Two AIRs deployed to cover all surfaces of an I-beam.

are deployed to perform the task of grit-blasting. At each assigned base placement of an AIR, a set of discrete points, $O_{ik} = \{o_{ik1}, o_{ik2}, \ldots, o_{ik(n_i^O)}\}$ with a radius r^o, are used to represent the surfaces of an object and are located inside the workspace boundary of the AIR where k is the base placement index of the ith AIR and n_i^O is the total number of discrete points that falls inside the workspace boundary of the ith AIR. These discrete points will henceforth be referred to as *targets*. The definition of targets and the method to create targets were explained in detail in Section 1.3. Due to constraints such as the joint angle limits of the AIR, some of these targets may be unreachable by the AIR. In order for a target, $\boldsymbol{o}_{ikj} \in O_{ik}$ to be *reachable* by the ith AIR, a feasible AIR pose $\boldsymbol{q}_{ikj}^f = [\theta_1, \theta_2, \ldots, \theta_{n^J}]$ needs to be found for the target where θ_1 to θ_{n^J} are the angles of the n^J AIR joints.

A feasible AIR pose is one that can reach the target with appropriate end-effector orientation and position, and without collision, e.g. the pose of AIR 1 ($i = 1$) shown in Fig. 4.4a which is generated to cover the target $\boldsymbol{o}_{ikj}$. One option for computing a feasible AIR pose for a target is to use the lookup table explained in Appendix C. Another option is to perform inverse kinematics using a numeric approach (e.g. optimization-based) or an analytical approach (if possible) and then performing collision checking to assess feasibility. Determining a feasible AIR pose should account for minimal torque on the AIR's joints and maximal manipulability measure, e.g. as explained in Appendix C, so as to calculate the following objectives more accurately.

At each of the assigned base placements of an AIR, a path can link the centroid of the reachable targets together using a single robot coverage path planning algorithm [5]. Ideally, the generated paths of the AIRs at all the selected base placements result in complete coverage paths, such as the paths shown in Fig. 4.4b, which cover all surfaces of the object. Thus, to achieve complete surface coverage, the base placements of all AIRs should be selected such that all targets representing the surfaces can be reachable by feasible poses of the AIRs. The size of the targets shown in Fig. 4.4a are the same for both AIRs, since in this example both AIRs are identical. However, depending on the capacity of each AIR, the size and the density of the targets can be different, meaning that the paths generated on the surfaces of an object can also be different for each AIR.

This objective is to maximize the number of targets that can be reached by all AIRs. That is, to minimize missed-coverage:

$$\min_{Z} \mathrm{F}_1(Z) = 1 - \sum_{i=1}^{n} \frac{\sum_{k=1}^{n_i^v} \mathrm{N}^f(Z_{ik})}{n_i^T} \tag{4.1}$$

where Z is a set containing all the design variables that have a value greater than zero, n is the number of AIRs, n_i^v is the total number of base placements that the ith AIR needs to visit, n_i^T is the total number of targets associated with the ith AIR which represent all surfaces, and $\mathrm{N}^f(Z_{ik})$ calculates the number of targets that can be reached with feasible AIR poses at a base placement decided based on Z_{ik} which is a design variable in Z associated with the ith AIR and its value determines the FBP to be visited by the ith AIR for the kth base placement. Since the AIRs can have different capacities, then each AIR can be associated with a different set of targets that represent the surfaces. Hence, n_i^T can be different for each AIR (i.e. for each i). The overlapped targets, which more than one AIR can reach, need to be counted only once. The overlapped targets can be found by performing a simple distance query and can be partitioned and allocated based on the APA approach presented in Chapter 3 or based on the "first come, first served" basis. Note that in Eq. (4.1), $\mathrm{F}_1(Z) = [0, 1]$ and represents the percentage of missed-coverage. A value of 0 for $\mathrm{F}_1(Z)$ corresponds to an optimal result, meaning that there is no missed-coverage and all areas of interest are covered. Conversely, a value of 1 corresponds to the worst possible result, meaning that the AIRs could not cover any section of the surface.

Objective 2 - Minimal Makespan: The second objective is to minimize the makespan (i.e. the overall completion time of the task). Optimizing this objective has the added benefit of equitably dividing the workload between the AIRs, because in order to achieve the minimal makespan, the coverage task needs to be divided appropriately amongst the AIRs. This objective also takes into account the set-up time associated with repositioning an AIR. If the cost of repositioning is set appropriately, then minimizing the makespan can also minimize the number of base placements needed. Thus, this objective is to minimize the makespan:

$$\min_{Z} \mathrm{F}_2(Z) = \max\{\mathrm{T}_1(Z), \mathrm{T}_2(Z), \ldots, \mathrm{T}_n(Z)\} \tag{4.2}$$

where $\mathrm{T}_i(Z)$ gives the completion time of the ith AIR, which can be calculated as

$$\mathrm{T}_i(Z) = \left(\sum_{k=1}^{n_i^v} \mathrm{N}^f(Z_{ik}) \cdot \frac{d_i}{v_i} \right) + n_i^v \cdot t_i^s \tag{4.3}$$

where n_i^v is the total number of base placements that the ith AIR needs to visit, $\mathrm{N}^f(Z_{ik})$ calculates the number of targets that can be reached with feasible AIR poses at a base placement decided based on the value of Z_{ik} in Z, d_i is the distance between two adjacent targets along a path of the ith AIR, v_i is the chosen end-effector speed of the ith AIR suitable for the application, and t_i^s is the set-up time associated with the ith AIR moving to the next base placement, which may include tasks such as turning off/on accessories and tools.

Note that this objective (minimal makespan) is in conflict with the Objective 1 (maximal coverage in Eq. (4.1)). The conflict is due to the following reasons: (i) the aim of Objective 1 is to minimize missed-coverage by increasing the number of targets that each AIR covers which in turn increases the makespan, hence Objective 1 is optimal when all areas are covered whereas Objective 2 is optimal when no area is covered, and (ii) Objective 1 has the potential to cause a larger number of FBPs to be selected for each AIR so as to maximize coverage by reaching more targets, whereas Objective 2 aims to minimize the number of FBPs to be selected for each AIR since the second term in Eq. (4.3) (after the plus sign) considers a penalty to account for the set-up time. Due to the conflict in these objectives and the following objectives, a multi-objective optimization algorithm is found to be appropriate for solving the optimization problem.

Equation (4.3) considers constant end-effector speed. An alternative to Eq. (4.3) is shown below (Eq. (4.4)). In certain situations, non-uniform coverage may be required, e.g. when grit-blasting a steel object where only certain areas of the object are heavily rusted. As expressed in Eq. (4.4), this area-specific focus can be accomplished by reducing the end-effector speed for such target areas. Therefore,

$$\mathrm{T}_i(Z) = \left(\sum_{k=1}^{n_i^v} \sum_{j=1}^{\mathrm{N}^f(Z_{ik})} \frac{d_i}{(w_{ikj} \cdot v_i^d) + v_i^{min}} \right) + n_i^v \cdot t_i^s \tag{4.4}$$

where $v_i^d = v_i^{max} - v_i^{min}$, and where v_i^{max} and v_i^{min} are the maximum and minimum end-effector speeds of the ith AIR, respectively, and w_{ikj} is the weighting factor, $0 \leq w_{ikj} \leq 1$, applied to the end-effector speed based on the condition of the area in which the target $\boldsymbol{o}_{ikj}$ is located (e.g. the level of rust). The smaller the value of w_{ikj}, the longer the period of grit-blasting applied to the target $\boldsymbol{o}_{ikj}$, and vice versa.

Objective 3 - Maximal Manipulability Measure: Performance metrics [131] such as the manipulability measure, the dexterity index, the minimum singular value, and measures of isotropy can be used to help obtain a measure for a manipulator or a manipulator pose corresponding to a certain point in the workspace. The use, limitations, and benefits of each of these measures depend on the application and the structure of the system or the robot manipulator. Manipulability measure [130] can be used to obtain a measure for a manipulator pose. The higher the sum of the manipulability measures of the poses for the targets, the higher the likelihood of finding a feasible trajectory during the task execution, since a large manipulability measure of a pose corresponds to a pose that is far away from singularities and that can move more freely in all directions. Therefore, this objective is to maximize the sum of manipulability measures for all AIR poses corresponding to all targets representing the surfaces. That is,

$$\max_{Z} \mathrm{F}_3(Z) = \sum_{i=1}^{n} \sum_{k=1}^{n_i^v} \sum_{j=1}^{\mathrm{N}^f(Z_{ik})} \mathrm{W}(\boldsymbol{q}_{ikj}^f) \tag{4.5}$$

where

$$\mathrm{W}(\boldsymbol{q}_{ikj}^f) = \sqrt{\det\left(\mathbf{J}(\boldsymbol{q}_{ikj}^f)\mathbf{J}^{\mathsf{T}}(\boldsymbol{q}_{ikj}^f)\right)} \tag{4.6}$$

is the manipulability measure (a value from 0 to 1), n_i^v is the total number of base placements that the ith AIR needs to visit, $\mathrm{N}^f(Z_{ik})$ calculates the number of targets that can be reached with feasible AIR poses at a base placement decided based on the value of design variable Z_{ik} in Z, $\mathbf{J}(\boldsymbol{q}_{ikj}^f)$ is the Jacobian of the pose $\boldsymbol{q}_{ikj}^f$. When determining a feasible pose for an AIR, maximizing manipulability measure for the AIR needs to be considered, e.g. as per the lookup table in Appendix C.

This objective (maximal manipulability) is in conflict with Objective 2 (minimal makespan). Objective 2 aims to minimize the makespan which indirectly minimizes the number of targets to be covered and the number of FBPs for each AIR to operate from. On the contrary, this objective has the potential to achieve a greater value for $\mathrm{F}_3(Z)$ (in Eq. (4.5)), i.e. a greater sum of manipulability measures, when more targets are covered and possibly when the AIRs operate from a larger number of FBPs.

This objective (maximal manipulability) and Objective 1 (maximal coverage) benefit from covering a greater number of targets. However, greater coverage does not necessarily equate to a greater sum of manipulability measures. Hence, this objective is required. For example, there may be a number of base placements from which an AIR can cover the same targets; however, from one of these base placements, the AIR may be able to reach the targets with better manipulability measure. That is, there can be multiple solutions for which coverage is maximal, but not all solutions are the same in terms of manipulability measure, and vice versa. Objective 3 and Objective 1 may be combined using the weighted sum method which requires proper normalization of the objectives. In this case, Objective 1 should be given a greater weight since maximizing coverage is more important in complete coverage tasks. However, due to the suitability of multi-objective optimization with respect to other objectives, it is preferable for Objectives 1 and 3 not to be combined to enable a straightforward selection from, and comparison between, the Pareto optimal solutions (i.e. the output of the optimization).

Objective 4 - Minimal Torque: To improve the operating condition of an AIR, it is preferable to minimize the torque experienced by the joints of the AIR.

Let the torque ratio of joint m of a feasible AIR pose $\boldsymbol{q}^f_{ikj}$ be the amount of torque the mth joint has experienced divided by its torque capacity. The maximum torque ratio $\mathcal{T}^{Rmax}(\boldsymbol{q}^f_{ikj})$ for the pose $\boldsymbol{q}^f_{ikj}$ is the largest torque ratio from all the joints of the ith AIR, i.e.

$$\mathcal{T}^{Rmax}(\boldsymbol{q}^f_{ikj}) = \max_m \left| \frac{\mathcal{T}_{im}(\boldsymbol{q}^f_{ikj})}{\tau^c_{im}} \right| \tag{4.7}$$

where $\mathcal{T}_{im}(\boldsymbol{q}^f_{ikj})$ is the torque experienced by joint m of the ith AIR at pose $\boldsymbol{q}^f_{ikj}$, and τ^c_{im} is the torque capacity of the mth joint.

This objective is to minimize the sum of maximum torque ratios of the AIR that experiences the most amount of torque. That is,

$$\min_{Z} \mathrm{F}_4(Z) = \max_{i} \left(\sum_{k=1}^{n_i^v} \sum_{j=1}^{\mathrm{N}^f(Z_{ik})} \mathcal{T}^{Rmax}(\boldsymbol{q}_{ikj}^f) \right) \tag{4.8}$$

where n_i^v is the total number of base placements that the ith AIR needs to visit, and $\mathrm{N}^f(Z_{ik})$ calculates the number of targets that can be reached with feasible AIR poses based on the design variable Z_{ik} in Z.

For a feasible AIR pose $\boldsymbol{q}_{ikj}^f$, the torque at each joint is calculated by considering: (i) the weight of the nozzle, joints (actuators), and links of the AIR; and (ii) the reaction force generated on the nozzle, e.g. due to the stream of grit or paint exiting the nozzle. Readers are advised to refer to Appendix B for information on calculating torque for an AIR manipulator pose. When finding a feasible pose for an AIR at a base placement, the feasible AIR pose needs to be determined such that the torque on AIR's joints is minimized, e.g. using the lookup table explained in Appendix C. Note that in the applications under consideration, such as grit-blasting and spray painting, the AIR moves at a slow speed when operating on a surface. Hence, torque due to angular, centripetal, and Coriolis accelerations [129] can be neglected.

This objective (minimal torque) is in conflict with Objective 1 (maximal coverage) and Objective 3 (maximal manipulability). Both Objectives 1 and 3 benefit from covering a larger number of targets, whereas similar to Objective 2 (minimal makespan), this objective (minimal torque) is negatively affected by larger coverage since the sum of torques experienced by the AIRs increases as coverage increases. It may seem then that this objective can be combined with Objective 2 (minimal makespan) since they both benefit from lower coverage. However, this objective can in fact be in conflict with Objective 2. In this objective (Objective 4), selecting a larger number of base placements from which the AIRs can operate may result in less torque to be experienced by the AIRs, whereas in Objective 2 there is a set-up time penalty proportional to the selected number of base placements so as to minimize makespan.

4.2.2.3 Constraint Functions

Keeping a safe distance between AIRs and relative to the objects are considered as the constraints.

Constraint 1 - Distance Between Any Two AIRs: The proximity of the AIRs with respect to each other at any time during the task execution should not be allowed to cause restrictions on their maneuverability or cause a high risk of collision. A minimum acceptable distance between the AIRs or a threshold δ should be determined based on the application or the structure of the AIRs. For example, for the AIRs shown in Fig. 4.4, δ can be the distance from an AIR's base to the workspace boundary of the AIR. For simplicity, the boundary can be approximated to be a sphere. Therefore, the AIRs' base placement should be chosen such that

$$\left\|\boldsymbol{\beta}_i^{AIR}(t) - \boldsymbol{\beta}_l^{AIR}(t)\right\| > \delta \tag{4.9}$$

$\forall i, l : i = 1, \ldots, n, l = 1, \ldots, n, i \neq l$, and $\forall t : t = 0, \ldots, t^c$ where $\boldsymbol{\beta}_i^{AIR}(t)$ and $\boldsymbol{\beta}_l^{AIR}(t)$ are the base placements of the ith and lth AIRs at time t, respectively, n is the total number of AIRs, and t^c is the overall completion time of the task.

An alternative option is to design this constraint as an objective function that maximizes the distance between the AIRs. Although this option may be helpful for some applications, it does not guarantee that it will meet the constraint of maintaining a safe distance between the AIRs. Thus, the solution needs to be checked for feasibility with respect to this constraint.

Constraint 2 - Distance to Obstacles: Note that the constraint on the distance between any AIR and the objects is already considered as part of the selection of FBPs. Recall that during the selection of FBPs, the base placements that are in close proximity to the objects are discarded.

4.2.3 Implementation of a Multi-objective Optimization Algorithm

An appropriate optimization algorithm needs to be utilized to test the proposed mathematical model. In this section, multi-objective Genetic Algorithm is first introduced along with a justification for why it is suitable for the problem under consideration. Then, for the specific problem being considered, the construction of the chromosomes, the crossover operator and the mutation operator, which are essential components of Genetic Algorithm, are detailed. The requirements and constraints are also discussed.

A group of algorithms that can be used for the presented mathematical model is the multi-objective optimization algorithms [135]. This group of algorithms is particularly useful for simultaneously optimizing multiple objectives that can be in conflict with each other, e.g. there can be circumstances where further optimizing an objective can only be done at the cost of weakening another objective or multiple other objectives. As a result, the output of the multi-objective optimization algorithms is a set of Pareto optimal solutions, rather than a single optimal solution. This set of Pareto optimal solutions forms what is referred to as the Pareto front. An advantage of obtaining the Pareto front is that the strategy for selecting the final solution from the Pareto front can be conveniently modified to suit the changes that occur within the application, which may change the importance of different objectives, without the need for repeating the optimization.

Multi-Objective Evolutionary Algorithms (MOEAs) are a group of algorithms used for multi-objective optimization. For the last two decades, strong and continuing research has been dedicated to the development of evolutionary algorithms [128]. An example of MOEAs is Non-dominated Sorting Genetic Algorithm II (NSGA-II) [132] which can be a suitable option for the problem under consideration.

Metaheuristic algorithms such as NSGA-II can be helpful in addressing NP-hard problems, particularly for problems with high combinatorial complexity and discrete search space. The optimization problem under consideration is a combination of the well-known Art Gallery Problem (AGP) and the Multiple Traveling Salesmen Problem (MTSP) [36, 136], both of which are considered to be NP-hard. AGP asks for the minimum number of points (and their locations) from which the entire environment can be observed, which is, in the problem under consideration, the same as finding the minimum number of FBPs (and

their locations) from which the environment can be covered. In the MTSP, the goal is to find the visiting sequence of a number of cities by multiple salesmen such that the total cost (e.g. travel time) is minimized and constraints are satisfied. This is similar to finding the visiting sequence of the selected FBPs for each AIR such that AIR team's objectives and constraints are met. Thus, NSGA-II is suitable for the problem under consideration, and its effectiveness is verified using experimental results presented later in this chapter. Note that comparing NSGA-II to other metaheuristic optimization algorithms is outside the scope of this thesis.

The following are some general considerations when implementing the optimization for the problem under consideration. An example implementation of the fitness function is presented later in the chapter.

1. AIRs can be mounted on a mobile platform or can be manually moved from one base placement to the next (e.g. using mechanical means). Thus, each AIR can move to its next assigned base placement upon completing the task on its current base placement. That is, in deciding to go to the next assigned base placement, the AIR is not required to wait for any of the other AIRs to complete the task on its current base placement.

2. At all times, the constraint of maintaining a minimum distance between any two AIRs mentioned in Section 4.2.2.3 must be met. Since any AIR can freely move to its next assigned base placement, then this constraint is temporal and does not necessarily mean that the distance between two AIRs at their kth assigned base placements is to be less than the threshold δ (used in Eq. (4.9)). For example, one AIR can be operating on its fifth base placement while another AIR can still be on its second base placement.

3. It is possible for the workspace of several AIRs to overlap at certain base placements. As a result, some of the targets in the overlapped workspace can be reachable by more than one AIR. These overlapped targets can be divided amongst the AIRs (e.g. using the APA approach presented in Chapter 3), or can be allocated to one of the AIRs based on the "first come, first served" strategy.

4.2.3.1 Chromosome Representation

A chromosome representation is developed that is designed and tested specifically for the problem under consideration. The work in [136] uses a two-part chromosome representation to solve the Multiple Traveling Salesmen Problem (MTSP), and the authors explain that the two-part chromosome reduces the search space when compared to one-chromosome and two-chromosome representations. The developed chromosome representation is to a certain extent similar to the two-part chromosome [136]; however, there are two main differences. As shown in Fig. 4.5, the first difference is that instead of having two parts for each chromosome, multiple parts are considered where each part is associated with one of the AIRs. The second difference is that instead of having a part in the chromosome dedicated to representing the required number of base placements for each AIR, i.e. for determining the value of $n_i^v, \forall i : i = 1, \ldots, n$, this parameter is made to correspond to the fixed length of the ith part of the chromosome associated with the ith AIR.

Fig. 4.5 shows an example of the developed chromosome representation where the first part of the chromosome is associated with the first AIR and has a length of three. That is, there are three nonzero genes in this part, which represent the indices of the FBPs that the first AIR needs to visit. The visiting sequence of the selected FBPs is from left to right. Therefore, in this example, AIR 1 first visits the 3rd FBP followed by the 6th FBP and then the 11th FBP. The same logic follows for AIRs 2 to n.

Alternatively, each part of the chromosome can be made with a length equaling the number of FBPs of the corresponding AIR, and binary encoding can then be used to determine the FBPs that need to be visited by the AIR. However, to reduce the length of the chromosome and also to reduce the search space, each part is generated with a length based on the capacity (e.g. workspace size) of the corresponding AIR. For example, for two identical AIRs with the same capacity, if either AIR can individually cover the entire object using

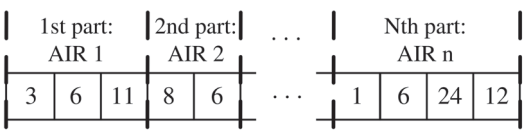


Fig. 4.5: A multi-part chromosome representation developed for the problem under consideration.

a minimum of 8 FBPs, then the length of each part associated with each of the two AIRs is 4 (i.e. $8/2 = 4$). Meaning that if 8 appropriate FBPs are chosen for both AIRs (4 for each AIR), then the entire object is covered. If the AIRs' capacities are different, then the length of each part of the chromosome can take into account the capacity of the AIRs. In Fig. 4.5, the length of the first, second and nth part of the chromosome is 3, 2 and 4, respectively, meaning that AIR 2 has a greater capacity (e.g. larger workspace size) than AIR 1, and AIR 1 has greater capacity than AIR n. Besides reducing the search space and chromosomes' length, another advantage of this chromosome representation is that when additional base placements are considered for finer discretization of the search space (explained in Section 4.2.1), the length of the chromosome doesn't need to be changed. This is because the number of base placements to be visited by each AIR will remain the same.

It may not be possible to accurately determine the number of base placements n^v needed for all AIRs to collectively cover the entire object. Thus, a reasonable approximation can be used based on the size of the object, or the number and the density of the targets that represent the object. The initial population can be generated such that for each part of a chromosome there are additional genes with a value of zero, as shown in Fig. 4.6. When creating a new population at each GA iteration, chromosomes with a different number of zero genes can be made, e.g. through crossover operation. The genes with the value of zero are interpreted as void (i.e. they don't represent any base placement). Thus, incorporating additional genes provides the flexibility to increase or reduce the number of FBPs to be visited by each AIR. The greater the uncertainty in approximating n^v for an application or an environment, then the larger the number of zero genes that can be added to the chromosome when generating the initial population. Note that the requirements that will be outlined in the following explanation of the crossover and mutation operators need to be accounted for so as to prevent infeasible solutions being added to the initial population.

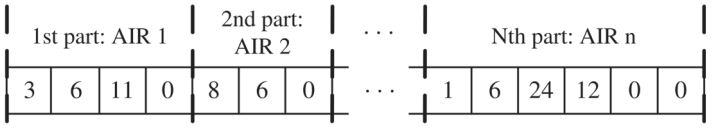


Fig. 4.6: Multi-part chromosome representation with additional zero genes to deal with the uncertainty in determining the number of base placements to be visited by the AIRs.

4.2.3.2 Crossover Operator

An example crossover operator is presented in this section, and the requirements that need to be taken into account for achieving feasible solutions are explained. An offspring generated from a crossover operator is only added to the subsequent population if it satisfies all the relevant requirements; otherwise, it is discarded. Alternatively, the crossover operator can be designed such that a generated offspring automatically meets the requirements.

In performing a crossover operation, a pair of parent chromosomes are selected, and a child chromosome is generated from the parents in the hope that the child chromosome will provide a better solution. There are several methods for selecting and exchanging genes from the parents chromosomes to form the child chromosome. Uniform crossover [137, 138] with some modifications is used for the developed chromosome representations; however, other crossover methods can also be used. Figure 4.7 is an example where crossover operation is performed on two selected parents to generate the child chromosome. There are three parts for each chromosome meaning that there are three AIRs. The length of each part of the chromosome is different since each AIR has a different capacity. It can be seen that for each part of the chromosome a random number of nonzero genes are first selected from the dad's chromosome, then a random number of genes are selected from the mom's chromosome, and finally, the rest of the genes are given a value of zero.

Several requirements need to be taken into account so as to obtain feasible solutions when designing the crossover operator for the problem under consideration:

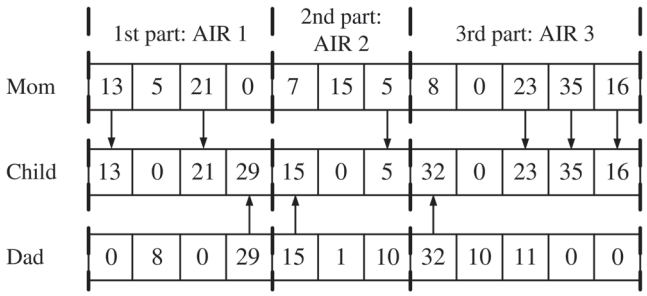


Fig. 4.7: An example of the crossover operation for the developed multi-part chromosome representation.

1. Let n_i^g be the number of genes in the ith part of a chromosome (i.e. the length) associated with the ith AIR. n_i^v nonzero genes are to be selected from the parents chromosomes for the ith part of a child chromosome such that $n_i^v \leq n_i^g$. n_i^v can be based on the n_i^v value of one of the parents chromosomes or can be a randomly generated value with the lower bound being the minimum number of base placements for the ith AIR. The remaining number of genes, i.e. $n_i^g - n_i^v$ genes, are assigned zeros.

2. n_i^D nonzero genes are selected from dad's chromosome and n_i^M nonzero genes are selected from mom's chromosome for the ith part of a child chromosome such that $n_i^D + n_i^M = n_i^v$. Selection of the genes from the dad's and mom's chromosomes can be random, left to right, etc. The selected nonzero genes can then be copied on the same genes of the child's chromosome, or be copied from left to right, randomly, etc.

3. Let g_{ik} be the kth nonzero gene in the ith part of a chromosome. $g_{ik} \neq g_{im}$, i.e. in the ith part of a chromosome, the kth nonzero gene and the mth nonzero gene cannot be the same since any AIR should not visit the same base placement more than once.

4. If the deployed AIRs are identical, then it should not be possible to have $g_{ik} = g_{jm}$, i.e. it should not be allowed to have the kth nonzero gene in the ith part of a chromosome be the same as the mth nonzero gene in the jth part of a chromosome. Having more than one identical AIR visiting the same base placement will not improve the result. However, if different AIRs are deployed, then each AIR would have a different capacity, and there can be potential for better performance or a greater coverage by another AIR visiting the same base placement.

4.2.3.3 Mutation Operator

An example mutation operator is presented in this section, and the requirements that need to be taken into account for achieving feasible solutions are explained. The purpose of the mutation operator is to maintain diversity from one population to the next, and this is done by altering the values of a small number of genes in a chromosome [137]. For the problem under consideration, only the nonzero genes are allowed to be altered so as to

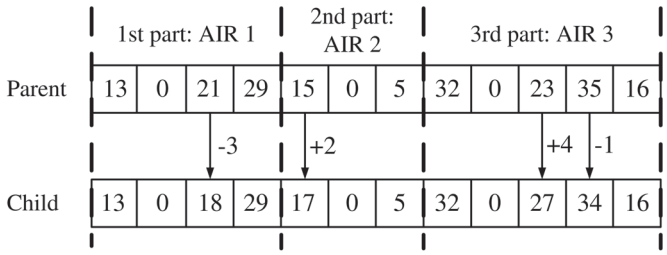


Fig. 4.8: An example of the mutation operation where small positive or negative random integers are added to a small number of genes.

avoid increasing or reducing the number of FBPs each AIR needs to visit as part of the mutation operation. The genes are altered by a small magnitude, and thus the aim is to potentially improve the performance of a chromosome by slightly changing the values (i.e. choosing neighboring FBPs) of a few nonzero genes. Figure 4.8 shows an example of the mutation operation where for each part, a random number of genes of the chromosome are slightly altered by adding a small positive or negative random integer to the genes.

The selection of the genes from the parent chromosome can, for example, be arbitrary. Note that points 3 and 4 of the requirements mentioned for the crossover operator (i.e. in Section 4.2.3.2) are also applicable to the mutation operator. A small value can be added to, or subtracted from, the kth selected nonzero gene of the ith part of the chromosome. In doing so, the lower and upper bounds are not to be violated, i.e. the following condition is to be kept true: $1 \leq g_{ik} + \delta^s_{ik} \leq n^F_i$ where δ^s_{ik} is a small negative or positive integer added to g_{ik}, and n^F_i is the number of FBPs associated with the ith AIR. This constraint prevents the mutated gene from having a value of zero (which means one less base placement would be visited), or to have a value greater than n^F_i.

4.3 Case Studies and Results

This section presents simulations and real-world experiments. It also includes studies on solution quality and consistency, validation of the simulation results using real AIRs, and comparisons between different solutions from the Pareto front. The first case study tests the OMBP method on three simulated objects which are separated from each other. Case Study 2 uses real data obtained from part of a steel bridge maintenance site, and the aim

is to test the OMBP method using three AIRs. The real-world experiment is conducted using two real AIRs and a vehicle that is targeted for grit-blasting. This experiment is a comparative study between the optimization result and the experiment result in terms of overall completion time and coverage, and is used to demonstrate that the OMBP method can be applied to real AIRs.

The function 'gamultiobj', which is based on NSGA-II, from the Matlab 2013 optimization toolbox was used for the experiments. Based on a brief investigation, it was found that the default parameters for Matlab 'gamultiobj' function perform well for the problem under consideration. Detailed investigation on the effects of different parameters, and comparing between the various optimization algorithms, is outside the scope of this thesis. The computer used to run the algorithm has the following specifications: 2.8GHz Intel Xeon E5-2680 v2 and 256GB 1866MHz ECC DDR3-RAM. However, the code is single threaded and hence only uses 1 core of the CPU at any one time.

The selection strategy to be designed for selection of a solution from the Pareto front is highly dependent on the application being considered. For the autonomous and one-off grit-blasting scenarios considered in the experiments, achieving an above threshold coverage is vital. Hence, a small subset of solutions is selected from the Pareto front such that an acceptable coverage percentage of the surfaces is obtained, i.e. narrowing down the solutions based on Objective 1. Note that for some applications 100 % coverage is necessary. Hence, the selection strategy in these applications must only select the solutions with 100 % coverage. From this subset of solutions, a further subset is chosen based on the makespan (Objective 2). Finally, the weighted average of the manipulability measure (Objective 3) and the torque (Objective 4) can be the basis of choosing the final solution from the reduced subset of solutions. Alternatively, the final solution can be selected based on improving the manipulability of the AIRs (i.e. Objective 3) or minimizing the torque experienced by the AIRs (i.e. Objective 4) if the joints condition of the AIRs is more critical.

The experiments consider the grit-blasting application, which is similar to many other surface preparation applications, such as spray painting, surface coating, and surface polishing. Please refer to Section 1.3 for a more detailed explanation of the grit-blasting

application.

Specifications of the AIRs used in the case studies are provided in Appendix A. The following values are used: (i) the constant end-effector speed of the AIRs is set to 0.1 m/s for the simulations, and 0.056 m/s for the real-world experiment, (ii) the size of the targets representing the objects is set to 0.04 m in radius, (iii) the overlap of two adjacent targets along a path is set to approx. 30 % of their diameter, (iv) the threshold δ used in Eq. (4.9) is set to 1 m, and (v) the set-up time t_i^s used in Eq. (4.3) is set to be 10 minutes. The first joint of the AIRs is a full 360° revolute joint. If the joint on the base of an AIR is not a full 360° revolute joint about the z-axis, then the AIR's base needs to rotate a number of times about the z-axis so as to cover all the reachable areas from a base placement.

As mentioned in Section 4.2.2.2, at a particular base placement, the ith AIR needs to find a feasible pose $\boldsymbol{q}_{ikj}^f$ for each target $\boldsymbol{o}_{ikj}$ that falls inside its workspace boundary. There are a number of ways for calculating a feasible AIR pose; however, in general, the process of finding a feasible pose for each target and at each base placement can be computationally expensive. Therefore, a lookup table (Appendix C) was used in the experiments to potentially reduce the computation time. The aim of finding feasible AIR poses for the targets is not to generate a trajectory for the AIR, but rather to determine which targets are reachable at a particular base placement. Thus, the use of a lookup table is effective and time efficient.

4.3.1 Procedure for Calculating the Objective Functions

Various multi-objective optimization algorithms can be used to solve the optimization problem. However, regardless of the optimization algorithm used, the objective functions are required to be computed iteratively within the optimization solver. Function 4.1 is shown and explained in this section so as to give an insight into the procedure used within the employed NSGA-II optimization algorithm for obtaining the values of the objectives. The presented fitness function is specific to the application being considered; however, its applicability can extend to a wide range of similar applications or scenarios. For example, for applications where a mobile or an automatic moving platform is not necessary or can add undesirable complications to the problem, an immobile AIR can be considered.

Immobile AIRs need to be repositioned manually by a human operator. Due to the safety risk of manually repositioning an AIR while another AIR is in operation, it is necessary to wait for all AIRs to complete their task at their current base placements prior to moving all AIRs. Small modifications to the fitness function can account for this change.

The inputs to the fitness function (Function 4.1) are the design variables Z and all targets, $O = \{O_i, \ldots, O_n\}$ associated with all AIRs. The design variables are encoded as the genes of the chromosomes used in the multi-objective GA. For example, the design variables associated with the ith AIR correspond to the nonzero genes in the ith part of the chromosome. The function loops n^v times (line 3) where n^v is the number of nonzero

Function 4.1 Objective Functions Evaluation.

1: **function FitFunc**(Z, O)
2: $k_i \leftarrow 1, \forall i : i = 1, \ldots, n$
3: **for** $counter^{base} = 1$ to n^v **do**
4: $[T^s, I^s] \leftarrow \text{Sort}(t_1, t_2, \ldots, t_n)$
5: $\acute{i} \leftarrow 1$
6: **while** $\acute{i} \leq n$ **do**
7: $i \leftarrow I^s_{\acute{i}}$
8: $k \leftarrow k_i$
9: **if** $k \leq n^v_i$ **then**
10: **if** $\|\boldsymbol{\beta}^{AIR}_i - \boldsymbol{\beta}^{AIR}_j\| \leq \delta$ **then**
11: $t_i \leftarrow t_j$
12: **end if**
13: $[O^{al}_i, \mathcal{T}^{al}_i, W^{al}_i, t_i] \leftarrow \text{Perf}(O, Z_{ik})$
14: $t_i \leftarrow t_i + t^s_i$
15: $k_i \leftarrow k_i + 1$
16: $\acute{i} \leftarrow n + 1$
17: **else if** $k_i > n^v_i$ **then**
18: $\acute{i} \leftarrow \acute{i} + 1$
19: **end if**
20: **end while**
21: **end for**
22: $\text{F}_1 \leftarrow \text{Coverage}(O^{al}, O)$
23: $\text{F}_2 \leftarrow \text{Makespan}(t_1, t_2, \ldots, t_n)$
24: $\text{F}_3 \leftarrow \text{Manipulability}(W^{al})$
25: $\text{F}_4 \leftarrow \text{Torque}(\mathcal{T}^{al})$
26: **return** $[\text{F}_1, \text{F}_2, \text{F}_3, \text{F}_4]$
27: **end function**

genes in a chromosome (i.e. the total number of base placements all AIRs need to visit). At each loop, the aim is to evaluate the performance (line 13) of an AIR at one of its assigned base placements. To do so, the progress times, $t_1, t_2, \ldots, t_n$, of the n AIRs are first sorted from the lowest to the highest value (line 4) where $T^s = \{T_1^s, T_2^s, \ldots, T_n^s\}$ with corresponding indices $I^s = \{I_1^s, I_2^s, \ldots, I_n^s\}$, and $T_i^s \in \{t_1, t_2, \ldots, t_n\}$ with corresponding index I_i^s. The progress time of an AIR is the time expected to have been taken by the AIR while carrying out the task up to its current base placement (i.e. current loop). Based on the sorted progress times T^s and corresponding indices I^s, the AIR with the minimal progress time (i.e. the ith AIR in line 7) moves to its next assigned base placement. However, only if the maximum number of base placements assigned to the AIR has not been exceeded (line 9) where n_i^v is the total number of base placements the ith AIR needs to visit. If the base placement $\boldsymbol{\beta}_i^{AIR}$ that the ith AIR will move to is close to another AIR's base placement $\boldsymbol{\beta}_j^{AIR} (j \in 1, \ldots, n, j \neq i)$ (i.e. the distance checking shown in line 10 where δ is the threshold used in Eq. (4.9)), then the ith AIR is made to wait until the jth AIR has completed the work at its current base placement (line 11). The fitness function is designed based on the "first come, first served" strategy, such that when an AIR moves to its next assigned base placement, it is allocated all the targets that it can reach and that are not yet allocated to another AIR. The performance at the next assigned base placement of the ith AIR is then evaluated (line 13 - further explained below) and the progress time t_i of the AIR is updated (line 14) by adding the set-up time t_i^s of the ith AIR to the progress time. In the case that the limit is reached in terms of the number of base placements to be visited by the ith AIR (line 17), then the next AIR with minimal progress time, i.e. $(i+1)$th AIR, is checked to be used (lines 18).

The output data obtained from the function Perf (in line 13) can be used to obtain the values of Objectives F_1 to F_4 (lines 22 to 25) based on Eqs. (4.1), (4.2), (4.5), and (4.8), respectively. For all the experiments, it is assumed that the AIRs need to obtain a uniform coverage of all surfaces, thus the completion time of an AIR is based on Eq. (4.3).

The function Perf (Function 4.2), which was used in line 13 of Function 4.1, is to evaluate the performance of an AIR at a particular base placement. The inputs to the function are the sets of targets in O, and Z_{ik} which is the design variable indicating the index of the FBP. Based on the design variable Z_{ik}, the targets O_{ik} that are inside the workspace boundary

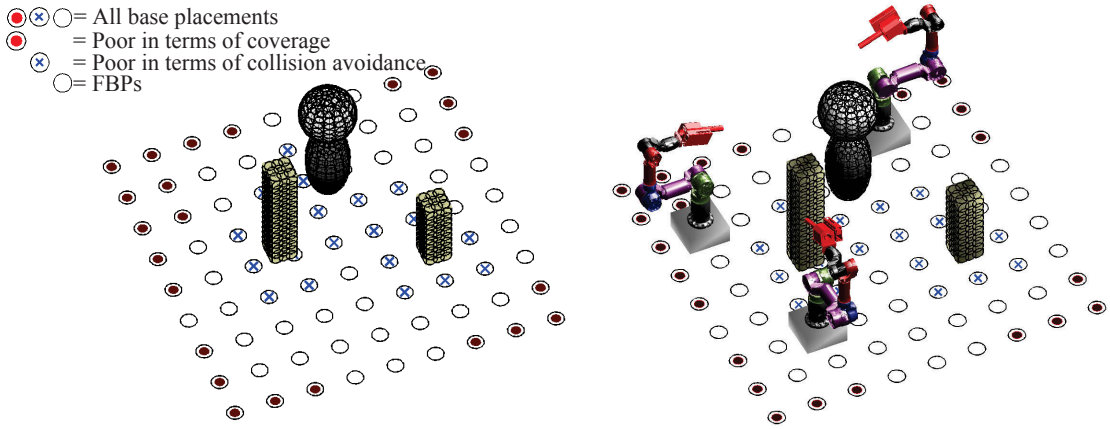
Function 4.2 Performance Evaluation at a Base Placement.

1: **function** PERF(O, Z_{ik})
2: **for** $counter^{rot} = 1$ to n^r **do**
3: **for** $j = 1$ to n_i^O **do**
4: **if** Reachable$(\boldsymbol{o}_{ikj}, Z_{ik}) = true$ **then**
5: $O_i^{al} \leftarrow O_i^{al} \frown \boldsymbol{o}_{ikj}$
6: $W_i^{al} \leftarrow W_i^{al} \frown \mathrm{W}(\boldsymbol{q}_{ikj}^f)$
7: $\mathcal{T}_i^{al} \leftarrow \mathcal{T}_i^{al} \frown \mathfrak{T}^{Rmax}(\boldsymbol{q}_{ikj}^f)$
8: $t_i \leftarrow t_i + t(\boldsymbol{o}_{ikj})$
9: **end if**
10: **end for**
11: **end for**
12: **return** $[O_i^{al}, \mathcal{T}_i^{al}, W_i^{al}, t_i]$
13: **end function**

of the ith AIR at the kth base placement can be obtained. An appropriate number of discrete base rotations (i.e. n^r in line 2 of Function 4.2), which an AIR needs to perform in order to cover all targets at a particular base placement, is to be predetermined based on the structure and the kinematics of the AIR. At the kth base placement, the function loops through the base rotations (line 2) and all targets $\boldsymbol{o}_{ikj} \in O_{ik}, \forall j : j = 1 \ldots n_i^O$ (line 3). For each target, if the target is reachable (line 4), i.e. if a feasible AIR pose $\boldsymbol{q}_{ikj}^f$ can be found for the target $\boldsymbol{o}_{ikj}$, then the target $\boldsymbol{o}_{ikj}$, the manipulability measure $\mathrm{W}(\boldsymbol{q}_{ikj}^f)$ (calculated based on Eq. (4.6)) due to the AIR's pose $\boldsymbol{q}_{ikj}^f$, and the maximum torque ratio $\mathfrak{T}^{Rmax}(\boldsymbol{q}_{ikj}^f)$ (calculated based on Eq. (4.7)) are added (lines 5 to 7) to the sets O_i^{al}, $\mathcal{T}_i^{al}$ and W_i^{al}, respectively. Note that the notation "$\frown$" represents concatenation. The superscript al is used to symbolize the *allocated* targets. The time it takes to cover the target $\boldsymbol{o}_{ikj}$ (i.e. $t(\boldsymbol{o}_{ikj})$) is also added to the progress time t of the ith AIR (line 8).

4.3.2 Case Study 1: Three AIRs Grit-blasting Three Objects

Three AIRs are used to grit-blast three objects which are separated from each other (unconnected). The three AIRs are identical, hence target representation (Fig. 4.9a) of the objects is the same for all three AIRs. The objects are represented using 664 targets. Out of the 81 discrete base placements shown in Fig. 4.9a, the 33 empty circles are the FBPs. The base placements with a cross are deemed to be too close to the objects and the



(a) All the base placements and the FBPs. **(b)** A solution chosen from the Pareto front.

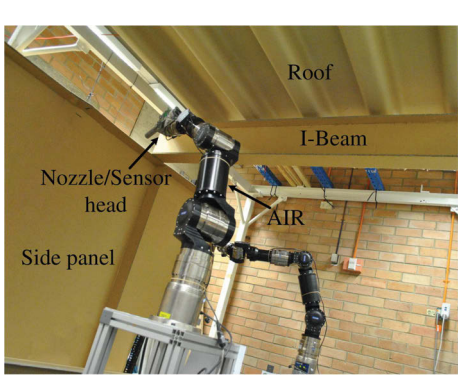
Fig. 4.9: Three AIRs select base placements to grit-blast three objects.

base placements filled with red circles have low coverage of the objects. For the particular application being considered, it was empirically found that the discrete base placements are best to be spaced at 0.3 m from each other. In this simulation, only Objectives 1 (maximal coverage) and Objective 2 (minimal makespan) are considered.

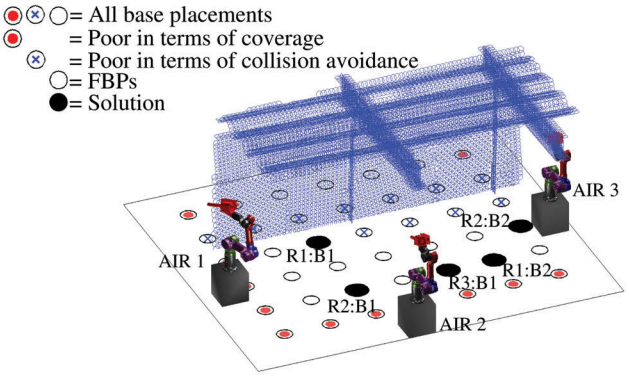
A solution chosen from the Pareto front is shown in Fig. 4.9b. Based on the Solution all 664 targets can be covered. The makespan is also optimal assuming that the AIRs can equitably partition and allocate the overlapped areas (e.g. using the APA approach presented in Chapter 3). Hence, since in this case study only one base placement is needed for each AIR, then the "first come, first served" strategy mentioned previously is not used for allocating the overlapped targets. The computation time for the case study is less than 5 minutes. To ensure that the optimization algorithm can produce consistent results every time it is run, the optimization process is repeated 10 times. The default maximum number of generations calculated by the optimization solver based on the number of design variables is 600; however, on average the optimization terminates at 105 generations. By default, Matlab calculates the population size based on the number of design variables, which is 40 for this case study.

4.3.3 Case Study 2: Three AIRs Applied in a Steel Bridge Maintenance Environment

A mock environment as shown in Fig. 4.10a is created using real data obtained from part of a steel bridge maintenance site. Simulations were then performed on the data obtained from the mock environment. Three identical simulated AIRs modeled upon real AIRs are used to perform the task of grit-blasting. The simulated scenario consists of 7518 targets (shown as small blue disks in Fig. 4.10b) to represent all the surfaces that are to be cleaned. The circles on the ground are all the discrete base placements, from which the 18 empty circles are the FBPs. When determining the FBPs, in order to discard the base placements that have low coverage of the targets representing the objects, an estimation for coverage can quickly be made by calculating the number of targets that fall inside the workspace boundary of an AIR at each of the discrete base placements. This estimation can significantly reduce the computation time since feasible AIR poses are not generated for assessing the reachability of the targets. However, after determining the FBPs and during the optimization process, an accurate measure of coverage need to be made based on the lookup table (Appendix C). Note that although only the internal surfaces of the structure need to be covered, there are discrete base placements generated at the rear of the structure, since some of the targets (such as the back-end of the roof) can only be reached from the rear.



(a) The environment in which scanning was performed to obtain the data.



(b) The discrete base placements, FBPs, target representation of the object, and the final result.

Fig. 4.10: The mock environment and its simulation, the location of the discrete base placements as well as the FBPs, and the result of the simulation.

Table 4.1: Three solutions from the Pareto front.

Soln. No.	Obj. 1: coverage (%)	Obj. 2: makespan (s)	Obj. 3: manipulability	Obj. 4: torque (N.m)
1	99	3328	-459	1991
2	98.4	3261	-435	2565
3	96.5	2505	-374	1724

Based on the designed selection strategy and assuming a coverage threshold of 96 % for this scenario, a solution from the Pareto front is selected. Figure 4.10b is based on the chosen solution from the Pareto front where the annotated and filled black circles are the selected base placements for the AIRs. The notation Ri:Bj represents the ith AIR at its jth base placement to visit. Based on the chosen solution, 96.5 % of the *reachable* targets can be covered. Solutions with over 99 % coverage can be obtained from the Pareto front; however, this is at the cost of a significant increase to the makespan.

In Table 4.1, three solutions from the Pareto front are presented. For clarity, solutions related to Objective 1 are presented as the coverage percentage of the *reachable* targets rather than missed-coverage. As shown in the table, 99 % coverage of the reachable targets is possible. However, assuming a coverage threshold of 96 % for this scenario, then solution 3 would be the final solution (Fig. 4.10b is based on this solution). Solution 3 has a significantly better makespan than the other two solutions. Of course, the selection strategy can be changed to suit the desired expectations for the application. As explained in Section 4.2.1, finer base placement discretization will be needed if the desired threshold is not met. It needs to be noted that a maximum of 93.5 % of the targets that represent the object can actually be covered based on a brute force search used to obtain the ground truth. The rest of the targets, representing areas such as the top and the bottom of the I-beam flange and the back end of the roof, cannot be covered regardless of the base placement of the AIRs (unless the AIRs are equipped with a base that can move vertically). Hence, 96.5 % of *reachable* targets correspond to 90.2 % coverage of the entire object. Note that the optimizer minimizes objectives functions; however, Objective 3 (F_3), which is related to manipulability measure, has to be maximized. Thus, the optimizer minimizes $-F_3$. The computation time for this case study is less than 16 minutes on average.

4.3.4 Case Study 3: Solution Quality and Consistency

Tuning of optimization parameters have been briefly investigated and the default parameters for Matlab's 'gamultiobj' function were found to perform well for the problem under consideration. However, comparing between different optimization algorithms and a detailed investigation in further tuning the optimization algorithm's parameters are left for future work. Henceforth is a demonstration that NSGA-II and the developed chromosome representation are suitable for the problem under consideration. Solution quality and consistency in obtaining acceptable solutions are checked by repeating the optimization process (10 times) for Case Study 2 and then evaluating the results.

Figure 4.11 shows the results for Objective 1 (percentage of missed coverage) and Objective 2 (makespan in seconds) of the 10 optimization runs. The solutions are selected based on the same selection strategy used before and all solutions satisfy the 96 % coverage threshold of the *reachable* targets. The average of the 10 solutions for Objectives 1 to 4 is 8.1 % missed-coverage (corresponds to 98 % coverage of the reachable targets), 2641 s, -417 and 1605 N.m, respectively. Figure 4.12 shows the boxplots of Objectives 1 and 2 for the 10 runs.

Figures 4.13a and 4.13b show the Pareto front for the first 2 optimization runs. It can be seen that there are a number of solutions that meet the 96 % coverage threshold (less than or equal to 10 % missed-coverage). Note that although the Pareto front is shown with respect to Objectives 1 and 2, the Pareto front actually considers all four objectives, hence

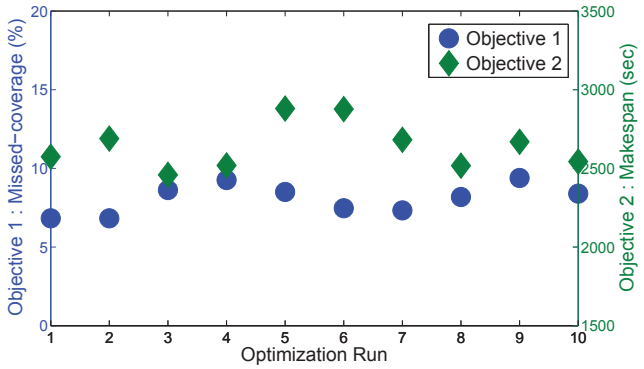
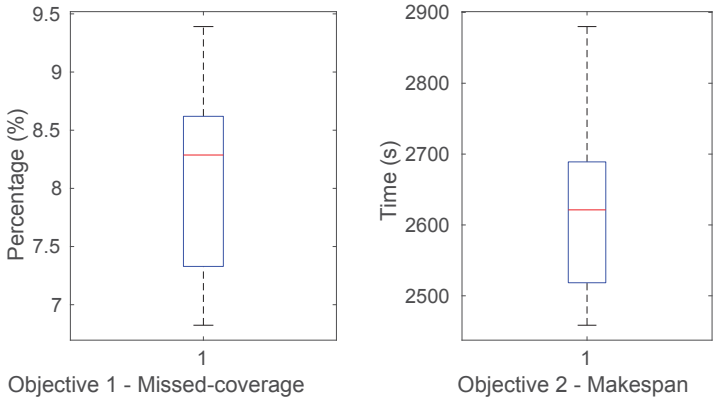
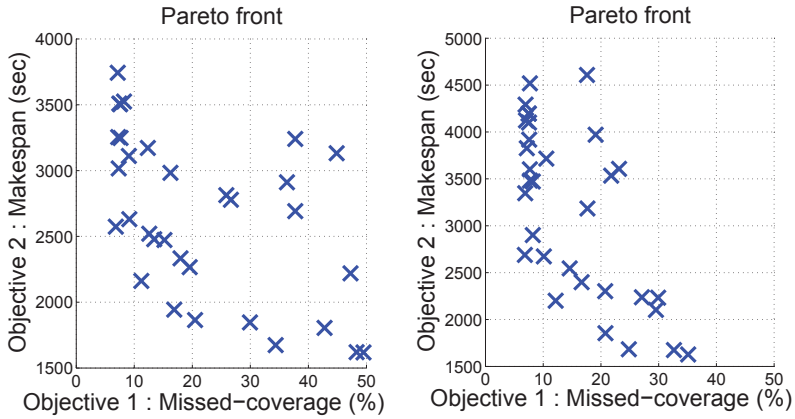


Fig. 4.11: Results for Objective 1 (percentage of missed coverage) and Objective 2 (makespan in seconds) of the 10 optimization runs.



(a) **(b)**

Fig. 4.12: Boxplots of Objectives 1 to 4 for the 10 optimization runs.



(a) Pareto front for the first optimization run.

(b) Pareto front for the second optimization run.

Fig. 4.13: Pareto fronts for the first two optimization runs with respect to Objectives 1 and 2 only.

some of the solutions are better in terms of Objectives 3 and/or 4 rather than Objectives 1 and 2. It was found that in all 10 optimization runs, the optimization terminates due to the average change in the spread of the Pareto front being less than the default tolerance set in 'gamultiobj' function of the Matlab optimization toolbox. The default maximum number of generations calculated by the optimization solver based on the number of design variables is 1200; however, on average the optimization terminates at 101 generations. By default, Matlab calculates the population size based on the number of design variables, which is 60 for this case study.

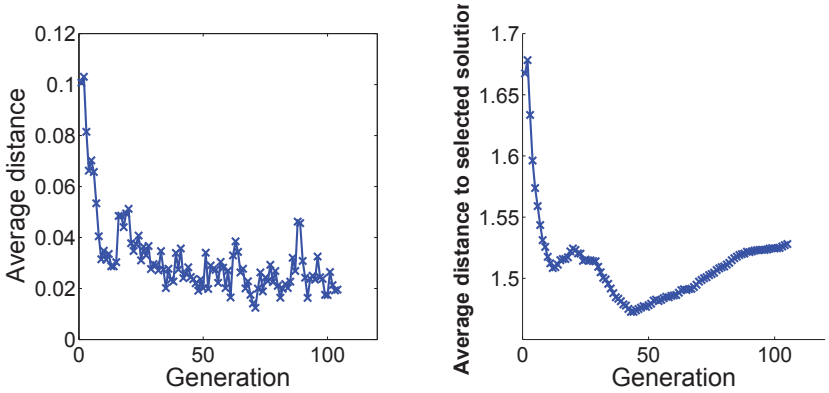


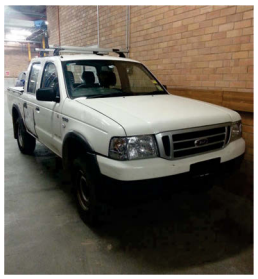
Fig. 4.14: (a) Average of distances of individuals at each generation; and (b) average of distances of all individuals in each generation to the selected solution.

Figures 4.14a and 4.14b are created from an optimization run. Figure 4.14a shows the average of distances of individuals at each generation using the field 'AverageDistance' of Matlab 'gamultiobj', which tends to favorably decrease as the generation number increases. At each generation, the average of distances of each member of the population to the nearest neighboring member is calculated and saved to the field 'AverageDistance'. Figure 4.14b shows the average of distances of all individuals in each generation to the selected solution from the Pareto front. The individuals in each generation were normalized with respect to the selected solution. It can be seen in Fig. 4.14b that for the first 45 generations, the average of distances decreases with respect to the selected solution. However, the average value then starts to slowly increase which could be due to the optimizer attempting to further diversify the population in the hope of arriving at better solutions. These solutions may also be better with respect to other solutions in the Pareto front.

The aim of investigating these parameters is to ensure consistency and to test that the solutions are appropriate for the problem under consideration; however, this information can be used for future studies as a means to investigate better/faster convergence (e.g. using a hybrid optimization) and to improve computational efficiency.

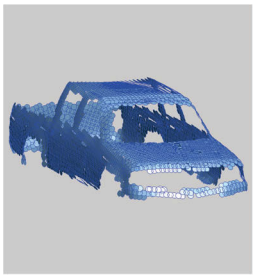
4.3.5 Case Study 4: Two AIRs Perform Grit-blasting on a Vehicle

As shown in Fig. 4.15, a real-world experiment using two AIRs and a vehicle (utility truck) is carried out. The experiment resembles the one-off task of grit-blasting of vehicles for removing old paint from metallic surfaces as a preparation for new paint. After scanning and processing the scan data, the point cloud that is shown in Fig. 4.15b can be obtained, which represents the metallic surfaces of the vehicle. The point cloud can then be used for target representation of the vehicle as shown in Fig. 4.15c where 3270 targets (shown as the blue disks) are used to represent the surfaces. In Fig. 4.15d, the circles on the ground are all the discrete base placements, from which the 87 empty circles (including the black filled circles) are the FBPs. The annotated and filled black circles are the chosen base placements for the AIRs where the notation Ri:Bj represents AIR i at its jth base placement to visit.

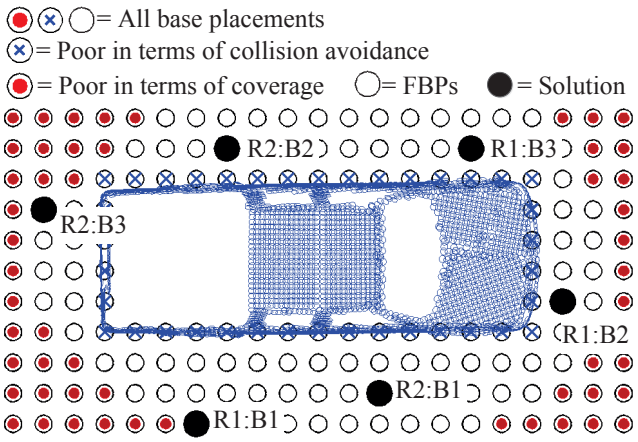


(a) A photo of the vehicle.

(b) Point cloud representation of the vehicle.



(c) Target representation of the vehicle.



(d) A solution chosen from the Pareto front.

Fig. 4.15: The vehicle, the point cloud representation and the target representation.

Solutions with over 99 % coverage can be obtained from the Pareto front. However, based on the designed selection strategy and assuming a coverage threshold of 90 % for this scenario, a solution from the Pareto front is selected with 94 % coverage of the reachable targets and a makespan of 3126 seconds. The difference between the completion times of the AIRs is only 46 seconds (less than 1.5 %). Figure 4.15d is based on the chosen solution. This solution provided values of 20 %, 3126 s, -186, and 507 N.m for Objectives 1 to 4, respectively. Note that 15 % of the targets, which mostly represent the roof of the vehicle, can't be covered by any AIR regardless of the chosen base placement since these targets are too high relative to the base of the AIRs. Thus, although the value of Objective 1 for the selected solution is 20 % (meaning that 20 % of the total targets were not covered), in reality only 6 % $(1 - 0.8/0.85)$ of the *reachable* targets were not covered. Figure 4.16 is created to illustrate the areas of the vehicle that are reachable based on the selected solution by showing the paths generated for both AIRs where the paths shown as solid blue lines and dashed red lines are associated with AIRs 1 and 2, respectively.

The feasibility of the obtained solution and the paths generated for both AIRs are checked using the AIRs. As an example, the path that is generated on the bonnet of the vehicle, which is associated with the first AIR at its second base placement, is shown in Fig. 4.17a. The experiment set-up at this base placement is shown in Fig. 4.17b. To check for the correct coverage of the path by the AIR, a laser is installed at the end-effector of the AIR and the laser point is tracked and compared to the simulated path. The highlighted targets in the path shown in Fig. 4.17a correspond to the laser point locations shown in

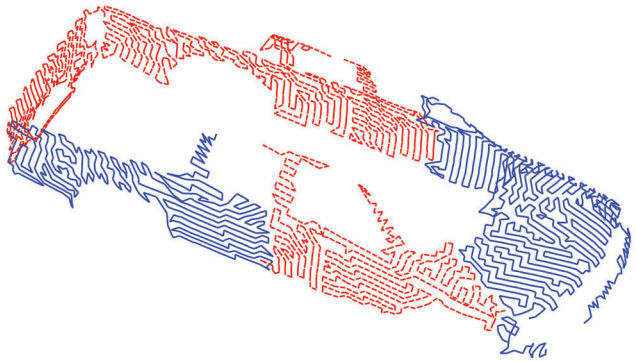
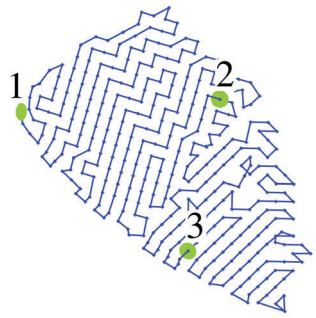


Fig. 4.16: Paths generated on all reachable surfaces of the vehicle, where the paths shown as solid blue lines and dashed red lines are associated with AIRs 1 and 2, respectively.



(a) A path generated on the bonnet of the vehicle and three highlighted targets.

(b) Set-up of the experiment.

(c) Laser point for the 1st highlighted target.

(d) Laser point for the 2nd highlighted target.

(e) Laser point for the 3rd highlighted target.

Fig. 4.17: The coverage of the path associated with the first AIR at its second base placement is checked using a laser that is installed at the end-effector of the AIR.

Fig. 4.17c to 4.17e.

To cover a slightly larger percentage of the reachable targets, the number of base placements to be visited by at least one of the AIRs needs to be increased, which would cause a significant increase in the completion time. In some practical circumstances, it can be more convenient and time efficient for the missed out sections to be covered manually by a human operator. Based on the selected solution, the completion time of AIRs 1 and 2 is expected to be 3088 s and 3126 s, respectively. Considering that the set-up time t_i^s per base for both AIRs is 10 minutes, and performing the experiment, the actual completion time of AIRs 1 and 2 is found to be 3080 s and 3111 s, respectively. It is sometimes not possible to generate paths that will appropriately and completely cover all reachable targets, and as a result, a small difference in completion times between the simulation result and the experiment can be present. However, the difference is insignificant (less than 15 s or 0.5 %) for this experiment.

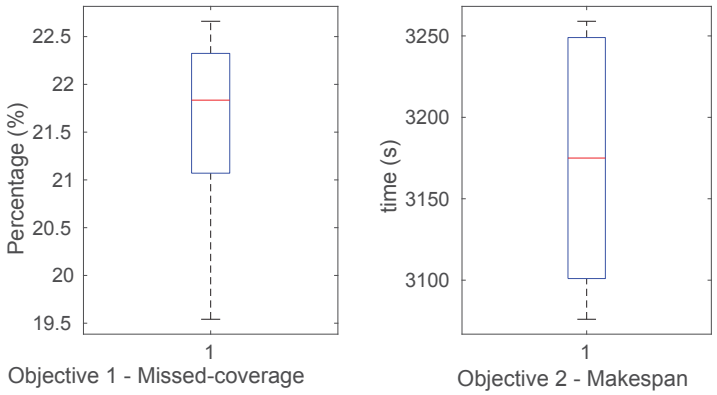


Fig. 4.18: Boxplots of Objectives 1 to 4 for the 10 optimization runs.

To ensure consistency in results and to test the optimization algorithm, the optimization process is repeated 10 times for this experiment using the same procedure outlined in Section 4.3.4. For each run, a solution from the Pareto front is chosen based on the above selection strategy. The average of the 10 solutions for Objectives 1 to 4 is 21.6 % (corresponds to 92.2 % coverage of reachable targets), 3170 s, -181 and 498 N.m, respectively. Figure 4.19 shows the boxplots of Objectives 1 and 2 for the 10 runs. Figure 4.19 shows the Pareto front for one of the optimization runs. Note that although the Pareto front is shown with respect to Objectives 1 and 2, the Pareto front actually considers all four objectives, hence some of the solutions are better in terms of Objectives 3 and/or 4 rather than Objectives 1 and 2. It was found that in all 10 optimization runs, the optimization terminates due to the average change in the spread of the Pareto front being less than the default tolerance set in 'gamultiobj' function of the Matlab optimization toolbox. The default maximum number of generations calculated by the optimization solver based on the number of design variables is 1200; however, on average the optimization terminates at 103 generations. By default, Matlab calculates the population size based on the number of design variables, which is 60 for this case study. The computation time for this case study is less than 6 minutes on average.

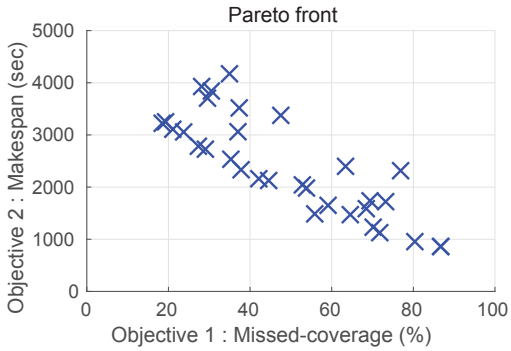


Fig. 4.19: Pareto front with respect to Objectives 1 and 2 only.

4.4 Discussion

All targets representing the surface of the three objects were covered in Case Study 1 (Section 4.3.2). In Case Study 2 (Section 4.3.3), for the steel bridge maintenance environment where three AIRs were deployed, on average, 98 % coverage of the reachable targets was achieved when assuming a 96 % coverage threshold. Similarly, for the experiment in Case Study 4 (Section 4.3.5) where two real AIRs were deployed to mimic the grit-blasting operation on a vehicle, on average, 92.2 % coverage of the reachable targets was achieved with near-optimal makespan when assuming 90 % coverage threshold. The optimization process was repeated 10 times for each case study to ensure consistency in achieving acceptable results, and to check the suitability of using NSGA-II and the developed chromosome representation for the problem under consideration. For the steel bridge maintenance environment (Case Study 2) and the vehicle (Case Study 4), a coverage percentage greater than 99 % could be achieved; however, this is at the cost of a substantial increase in the makespan. The significant increase in makespan to cover the small percentage of remaining areas is mainly due to the conservative value that is chosen for the setup time in Eq. (4.2). Hence, the solutions are selected from the Pareto front such that the coverage percentage satisfies the coverage threshold considered for the scenario while providing near-optimal makespan.

4.5 Conclusions

The OMBP method was presented in this chapter to address the problem of optimal base placements of AIRs for complete coverage tasks. The presented method enabled each AIR to simultaneously determine: (i) an appropriate number of base placements, (ii) the location of the base placements, and (iii) the visiting sequence of the chosen base placements, such that complete coverage of the surfaces is obtained. The method included search space discretization and candidate base placements selection, followed by a mathematical model that takes into account multiple objectives (i.e. maximal coverage, minimal makespan, maximal manipulability measure and minimal joints' torque). A minimum distance constraint between AIRs is enforced. Simulations and real-world experiments were conducted to compare the task specific objectives and to validate the method using different scenarios.

Chapter 5

A Stochastic Optimization-Based Method to Multi-AIR Base Placement for Complete Coverage Under Uncertainties

This chapter extends the method for Optimization of Multiple Base Placements (OMBP) presented in Chapter 4 by considering uncertainties in base placements due to sensing and localization errors. Thus, a method for Stochastic Optimization of Multiple Base Placements (Stochastic-OMBP) for each AIR, given the map of the objects, is proposed in this chapter. The differences between the Stochastic-OMBP method presented in this chapter and the OMBP method presented in the previous chapter are as follow: (i) the mathematical model in this chapter aims at achieving complete coverage under uncertainties in base placements, and (ii) the optimization problem in this chapter is solved using a multi-objective stochastic optimization algorithm which is based on a hybrid Genetic Algorithm (GA) and Simulated Annealing (SA) approach. Two case studies based on a real-world object and a complex simulated object are provided to demonstrate the effectiveness of the method in various scenarios, e.g. various levels of uncertainties, different number of AIRs, and AIRs with different capabilities.

For certain environments or applications, it may be possible to reduce uncertainties to a negligible level, e.g. by utilizing high-quality sensors. However, this work considers

applications where the cost of utilizing high-quality sensors is prohibitive or achieving precise localization is impractical due to the nature of the environment.

The main results of this chapter were previously included in the following paper: Hassan et al. [26].

The rest of the chapter is organized as follows. Section 5.1 defines the problem being addressed. Section 5.2 provides the methodology of the Stochastic-OMBP method which includes an overview (Section 5.2.1), the mathematical model (Section 5.2.2), and the hybrid multi-objective GA-SA approach (Section 5.2.3). The details of the experiments and results are presented in Section 5.3. Section 5.4 provides a brief discussion, and concluding remarks are stated in Section 5.5.

5.1 Problem Definition

An example of a large object is shown in Fig. 5.1. Assume that the complete coverage task of spray painting needs to be carried out on the object using two AIRs. That is, the two AIRs, which are equipped with spray painting nozzles, are required to collectively operate on all accessible surfaces of the objects. To achieve complete coverage, it is clear that each AIR is required to operate from multiple base placements. It is assumed that the base of the AIRs is fixed during the spray painting operation so as to achieve stability for precise and uniform coverage.

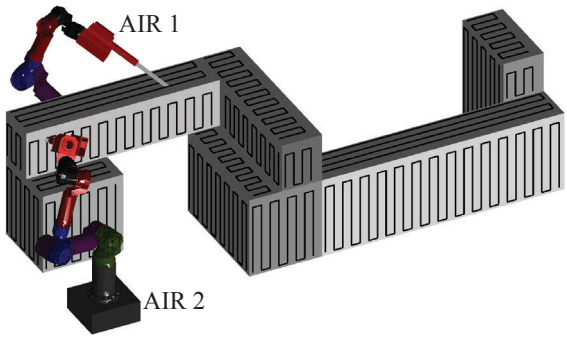


Fig. 5.1: Two AIRs spray painting a large object.

The problem addressed in this chapter can be concisely stated as follows: given probability distributions that represent the uncertainties in the AIRs' base placements, how would the AIRs select an optimal number and sequence of base placements such that complete coverage is achieved and the AIR team's objectives are optimized. The following team objectives are considered:

- Coverage of the objects must be complete or acceptable (above a threshold).
- Makespan should be minimal.
- Collisions must be avoided by maintaining safe distances from the objects and also between the AIRs.

The uncertainties in base placements of the AIRs can be due to various errors. Examples of the sources of uncertainties are: (i) errors in the collected sensor data (e.g. due to measurement inaccuracies and imprecision), and (ii) localization error (e.g. due to the limitations of the localization algorithm used with respect to the available features from the environment).

5.2 Methodology

5.2.1 The Stochastic-OMBP Method

The Stochastic-OMBP method is shown in Algorithm 5.1. The algorithm is similar to that of the previous chapter (Chapter 4) where the OMBP method was introduced. Thus, the algorithm is presented again to provide a summary of the main components, and also to explain the main difference between OMBP and Stochastic-OMBP. The Stochastic-OMBP first considers discretizing the search space (line 1). An example of the discretization is shown in Fig. 5.2. It can be seen that the density of the discrete base placements for each AIR is different because the AIRs' capacity (e.g. workspace size) is different.

The next step of the Stochastic-OMBP is to select a subset of discrete base placements for each AIR (line 2), called the Favored Base Placements (FBPs). FBPs are predicted to be superior in terms of both coverage and collision avoidance.

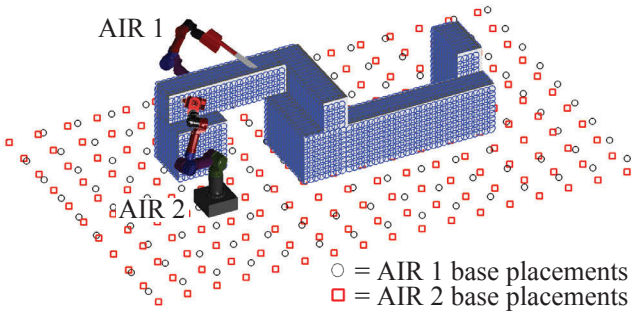


Fig. 5.2: Discrete base placements of two AIRs with different capacity, and target representation of the surfaces.

The main difference between the OMBP method (presented in Chapter 4) and the Stochastic-OMBP method lies in step 3 (line 3). The proposed mathematical model is solved using a multi-objective stochastic optimization (line 3), which is based on modifications of existing hybrid multi-objective algorithms. The hybrid algorithm uses Genetic Algorithm (GA) and Simulated Annealing (SA) algorithm and will be elaborated upon in Section 5.2.3. The mathematical model in the Stochastic-OMBP method is different to the OMBP method in that probability distributions representing uncertainties are taken into account. Since multi-objective optimization is considered, then the output of the optimization is the Pareto front (a set of Pareto-optimal solutions) from which a final solution is selected (line 4). After running the optimization, if a threshold (e.g. minimum coverage) is not reached (line 5), then finer discretization can be iteratively generated (line 6 and 7).

Algorithm 5.1 Stochastic-OMBP.

1: For each AIR, create n_i^b discrete base placements around the target objects ($\forall i : i = 1, 2, \ldots, n$)
2: Find favored base placements (FBPs)
3: Perform multi-objective optimization using hybrid GA-SA
4: Select the appropriate/best solution, Z^* from the Pareto front
5: **if** a threshold is not met **then**
6: Perform finer discretization around desirable FBPs
7: **goto** line 3
8: **end if**
9: **return** solution Z^* for execution

5.2.2 Mathematical Model

5.2.2.1 Design Variables

Let $B_i = \{\boldsymbol{b}_{i1}, \boldsymbol{b}_{i2}, \ldots, \boldsymbol{b}_{i(n_i^b)}\}$ be a set of candidate discrete base placements, and $B_i^{FBP} = \{\boldsymbol{\beta}_{i1}, \boldsymbol{\beta}_{i2}, \ldots, \boldsymbol{\beta}_{i(n_i^F)}\} \subseteq B_i$ be the FBPs for the ith AIR where $i = 1, 2, \ldots, n$. Each FBP can be defined as the x, y, z position coordinates. In the case when the rotation of the base about the z-axis effects the coverage performance of the AIR (e.g. if the joint on the base is not a 360° revolute joint), then the AIR's base needs to rotate a number of times about the z-axis so as to cover all the reachable areas from a base placement. The design variables are $Z_{ij} \in \{0, 1, \ldots, n_i^F\}$, $j = 1, 2, \ldots, n_i^v$, $n_i^v \leq n_i^F$, meaning that the jth base placement of the ith AIR is one of the n_i^F FBPs where n_i^v is an approximation of the number of FBPs to be visited by the ith AIR. As explained in the previous chapter (Section 4.2.3.1), an approximation of n_i^v can be made by considering the size of the object and its surface areas. The greater the uncertainty in approximating n_i^v, the larger its value. $Z_{ij} = 0$ means that the ith AIR will skip to the next, $(j+1)$th base placement decided by Z_{ij+1}.

5.2.2.2 Objective Functions

Two of the most important objectives in complete coverage tasks by multiple AIRs are: (i) achieving maximal coverage of the object, and (ii) completing the task with minimal makespan to improve productivity and cost efficiency. These objectives are formulated below. Additional objectives related to the performance of the AIRs and the intended task can be added if necessary.

Objective 1 - Maximal Coverage: For each AIR a set of *targets*, O_i are uniformly generated on all surfaces of the objects (e.g. the blue disks on the surfaces of the object shown in Fig. 5.2), with spacing and radius size that is chosen based on the capacity of the corresponding AIR. The definition of targets and the method to create targets were explained in detail in Section 1.3. At the jth base placement of the ith AIR, a subset of targets, $O_{ij} = \{\boldsymbol{o}_{ij1}, \boldsymbol{o}_{ij2}, \ldots, \boldsymbol{o}_{ijn^O}\} \subseteq O_i$ are located within the workspace of the AIR. Let $O_{ij}^r \subseteq O_{ij}$ be the *reachable* targets by the ith AIR at the jth base placement, and let

$O^u_{ij} \subseteq O^r_{ij}$ be the *uncovered* reachable targets (targets not yet covered). A path for the AIR to cover O^u_{ij} can be generated [5] by appropriately connecting the center points of the targets. In order for a target, $\boldsymbol{o}_{ijr} \in O_{ij}$ to be counted as reachable by the ith AIR, a feasible AIR pose $\boldsymbol{q}^f_{ijr}$ needs to be found that reaches the target $\boldsymbol{o}_{ijr}$ with an appropriate end-effector position and orientation (e.g. using the lookup table explained in Appendix C).

The aim of this objective is to minimize missed-coverage (areas not covered by any AIR). Due to the stochastic nature of the problem, the objective is formulated to minimize the expected value, i.e. :

$$\min_{Z} \mathrm{F}_1(Z) = E[\mathrm{F}^c(Z, \Xi^z)] \tag{5.1}$$

where Z is a set containing all the design variables, and each element in the set Ξ^z, i.e. $\Xi^z_{ij} \in \Xi^z$, is dependent on a design variable Z_{ij} and represents the uncertainties in the FBP, $\boldsymbol{\beta}_{i(Z_{ij})}$. Hence, $\Xi^z_{ij} \in \Xi^z$ is designed as a random vector with a multivariate normal distribution,

$$\Xi^z_{ij} \sim \mathcal{N}\big(\boldsymbol{\beta}_{i(Z_{ij})}, \boldsymbol{\Sigma}_i\big), \tag{5.2}$$

describing the x, y, z position errors associated with $\boldsymbol{\beta}_{i(Z_{ij})}$. The position error along the z-axis can be discarded for an AIR that can't move its base vertically. The function to calculate missed-coverage is expressed as:

$$\mathrm{F}^c(Z, \Xi^z) = 1 - \sum_{i=1}^{n} \frac{\sum_{j=1}^{n^v_i} \mathrm{N}^f\left(Z_{ij}, \boldsymbol{\xi}_{ij}\right)}{n^T_i} \tag{5.3}$$

where n^T_i is the number of targets in O_i, and $\mathrm{N}^f\left(Z_{ij}, \boldsymbol{\xi}_{ij}\right)$ calculates the number of uncovered reachable targets O^u_{ij} based on the design variable Z_{ij} and an independent observation $\boldsymbol{\xi}_{ij}$ of $\Xi^z_{ij} \in \Xi^z$.

Since the problem is stochastic, then an estimation of $\mathrm{F}_1(Z)$ (in Eq. (5.1)) can be found by using a number of independent simulations (i.e. Monte Carlo Simulations). For each $\Xi^z_{ij} \in \Xi^z$, K observations $(\boldsymbol{\xi}^1_{ij}, \boldsymbol{\xi}^2_{ij}, \ldots, \boldsymbol{\xi}^K_{ij})$ are generated. Let $(\Xi^z)_k$ be a set containing the kth observation from each $\Xi^z_{ij} \in \Xi^z$. An estimate of $\mathrm{F}_1(Z)$ is obtained using the sample

average:

$$\bar{\mathrm{F}}_1(Z) = \frac{1}{K}\sum_{k=1}^{K} \mathrm{F}^c\big(Z, (\Xi^z)_k\big) \tag{5.4}$$

where the observations to be taken from Ξ^z of Eq. (5.3) are the same observations as in $(\Xi^z)_k$.

Note that the targets located in the overlapped areas, i.e. areas that more than one AIR can reach, are only counted once. The overlapped areas can be allocated to an AIR based on a "first come, first served" strategy or based on the area partitioning and allocation approach presented in Chapter 3.

Objective 2 - Minimal Makespan: Another important objective is to minimize the makespan (i.e. the overall completion time of the task). The objective is formulated to minimize the expected value, i.e. :

$$\min_{Z} \mathrm{F}_2(Z) = E[\mathrm{F}^t(Z, \Xi^z)] \tag{5.5}$$

where

$$\mathrm{F}^t(Z, \Xi^z) = \max\left\{\mathrm{T}_1(Z, \Xi^z), \mathrm{T}_2(Z, \Xi^z), \ldots, \mathrm{T}_n(Z, \Xi^z)\right\}, \tag{5.6}$$

and the completion time of the ith AIR,

$$\mathrm{T}_i(Z, \Xi^z) = \left(\sum_{j=1}^{n_i^v} \mathrm{N}^f\left(Z_{ij}, \boldsymbol{\xi}_{ij}\right) \cdot \frac{d_i}{v_i}\right) + n_i^v \cdot t_i^s \tag{5.7}$$

where d_i is the distance between the center points of two adjacent targets that are on a path of the ith AIR, v_i is the speed that the end-effector of the ith AIR moves relative to a path, and t_i^s is the set-up time associated with switching off and on equipment/tools as the ith AIR navigates from one base placement to the next. Similar to $\mathrm{F}_1(Z)$, an estimation of $\mathrm{F}_2(Z)$ can be made using the sample average:

$$\bar{\mathrm{F}}_2(Z) = \frac{1}{K}\sum_{k=1}^{K} \mathrm{F}^t\big(Z, (\Xi^z)_k\big). \tag{5.8}$$

5.2.2.3 Constraint Functions (Distance Between Any Two AIRs)

At any time t of the task execution, the distance between any two AIRs must be greater than a threshold, δ so as to avoid collisions between AIRs and to allow each AIR's manipulator to maneuver more freely within its workspace. The constraint below needs to be satisfied:

$$\|\boldsymbol{\beta}_i^{AIR}(t) - \boldsymbol{\beta}_{\acute{j}}^{AIR}(t)\| - (e_i + e_{\acute{j}}) > \delta \tag{5.9}$$

$\forall i, \acute{j} : i = 1, 2, \ldots, n, \acute{j} = 1, 2, \ldots, n, i \neq \acute{j}$ where $\boldsymbol{\beta}_i^{AIR}(t)$ and $\boldsymbol{\beta}_{\acute{j}}^{AIR}(t)$ are the base placements from which the ith and the $\acute{j}$th AIRs are operating at time t, respectively. e_i and $e_{\acute{j}}$ are the maximum anticipated errors in base placements of the ith and $\acute{j}$th AIRs, respectively, which can be, for example, based on the limits of the 99.7 % confidence interval (3σ) of the normal distributions representing the uncertainties.

FBPs are the base placements that are a safe distance away from objects, hence a second constraint is not needed to keep a safe distance between any AIR and the objects in the environment.

5.2.3 Multi-objective Stochastic Optimization Approach

In order to solve and test the proposed mathematical model, multi-objective stochastic optimization based on hybrid GA-SA approach is used. It is inspired by the works in [139], [140] and [141]; however, modifications are made so as to appropriately address the problem under consideration.

The reason for using the hybrid GA-SA is the combined advantages of both the GA and the SA: (i) SA is "one of the fastest and universal probabilistic local procedures" [139]; (ii) for complex models, SA has been shown [141] to be ideally suited for stochastic optimization, and in fact even for non-stochastic models, artificially adding noise improves the performance of SA; (iii) GA can create a large number of diverse solutions at each generation; and (iv) selecting a neighbor solution in SA doesn't guarantee a better solution, however each generation in GA has a higher likelihood of producing better results than its preceding generation.

Algorithm 5.2 outlines the implementation of the hybrid multi-objective GA-SA for solving the stochastic optimization. Note that the parameters are defined at the top of the algorithm, some comments are added within the algorithm, and explanations of the lines in the algorithm are provided below. The algorithm takes advantage of utilizing the multinomial probability mass function [140] where an objective is randomly selected to become active in each iteration (i.e. by randomly selecting the value of $\acute{i}$ in line 12), and the varying sample size [141] where the number of observations, K gradually increases proportional to the GA iteration number, l (line 6).

Algorithm 5.2 Hybrid multi-objective GA-SA.

Input: Objective 1 & 2 initial temperatures: τ_1, τ_2, Cooling ratios: α_1, α_2, Initial population: $\boldsymbol{P}_0$, Fitness values for $\boldsymbol{P}_0$: $\mathrm{F}(\boldsymbol{P}_0)$, Population Size: n^p, No. of generations: n^{gen}, Max. No. of observations: n^K, No. of constant temperature loops: n^c

Output: Pareto-Optimal Solutions (POS)

1: **for** $l = 1$ to n^{gen} **do**
2: **for** $m = 1$ to n^p **do**
3: **for** $n^{cons} = 1$ to n^c **do**
4: $\acute{m}_{\acute{i}} \leftarrow \mathrm{random}\{1, 2, \ldots, n^p\}$, for $\acute{i} = 1, 2$, $\acute{m}_1 \neq \acute{m}_2$
5: $\mathcal{P}_{l,m} \leftarrow \mathrm{GenerateChild}\big(\mathcal{P}_{l-1,\acute{m}_1} \in \boldsymbol{P}_{l-1},\ \mathcal{P}_{l-1,\acute{m}_2} \in \boldsymbol{P}_{l-1}\big)$
6: $K \leftarrow \mathrm{ceil}\big(l \cdot n^K / n^{gen}\big)$
7: $\boldsymbol{\xi}^1_{ij}, \boldsymbol{\xi}^2_{ij}, \ldots, \boldsymbol{\xi}^K_{ij} \leftarrow \mathrm{GenerateObservations}\big(\Xi^z_{ij}\big),\ \forall i, j$
8: $\mathrm{F}(\mathcal{P}_{l,m}) \leftarrow [\bar{\mathrm{F}}_1(\mathcal{P}_{l,m}), \bar{\mathrm{F}}_2(\mathcal{P}_{l,m})]$ // based on Eqs. (5.4) & (5.8) where $Z = \mathcal{P}_{l,m}$
9: **if** $\mathrm{F}(\mathcal{P}_{l,m}) \succ \mathrm{F}(\mathcal{P}_{l-1,\acute{m}_1})$ **or** $\mathrm{F}(\mathcal{P}_{l,m}) \succ \mathrm{F}(\mathcal{P}_{l-1,\acute{m}_2})$ **then**
10: $\boldsymbol{P}_l \leftarrow \boldsymbol{P}_l \cup \mathcal{P}_{l,m}$ // add to $\boldsymbol{P}_l$ if $\mathcal{P}_{l,m}$ dominates a parent
11: **goto** line 18
12: **else if** $\mathrm{random}(0,1) < \exp\big(-\big(\bar{\mathrm{F}}_{\acute{i}}(\mathcal{P}_{l,m}) - \bar{\mathrm{F}}_{\acute{i}}(\mathcal{P}_{l-1,\acute{m}_{\acute{i}}})\big)/\tau_{\acute{i}}\big)$ where $\acute{i} = \mathrm{random}\{1,2\}$ **then**
13: $\boldsymbol{P}_l \leftarrow \boldsymbol{P}_l \cup \mathcal{P}_{l,m}$
14: **goto** line 18
15: **else if** $n^{cons} = n^c$ **then** // if no solution was added to $\boldsymbol{P}_l$
16: $\boldsymbol{P}_l \leftarrow \boldsymbol{P}_l \cup \mathcal{P}_{l-1,\acute{m}_{\acute{i}}}$ // accept a parent as a solution
17: **end if**
18: **end for**
19: **end for**
20: $[\boldsymbol{P}_l, \mathrm{POS}] \leftarrow \text{Update-NSGA-II}(\boldsymbol{P}_l, \boldsymbol{P}_{l-1}, \mathrm{POS})$
21: $\tau_1 \leftarrow \alpha_1 \tau_1, \quad \tau_2 \leftarrow \alpha_2 \tau_2$
22: **end for**
23: **return** POS

Algorithm 5.2 is designed with certain differences to the algorithms used in the aforementioned literature [139–141]. Instead of a single-objective GA, a controlled elitist NSGA-II [142], which is a multi-objective non-dominated sorting genetic algorithm, is used. An advantage of NSGA-II is that unlike the diversification strategy used in [139], the diversity in Algorithm 5.2 is achieved through NSGA-II itself [142] (i.e. when updating the current population in line 20). Additionally, since NSGA-II is a multi-objective optimization algorithm, then Pareto-Optimal Solutions (POS) can be obtained (line 23) and a solution can then be conveniently selected for the target application. However, POS need to be updated (line 20) after each generation. Termination criteria is also decided by NSGA-II rather than SA, e.g. if the maximum number of generations, n^{gen} is reached, or if a threshold on the average change in POS over a certain number of generations is met. As a result, since SA doesn't terminate the algorithm, then there is the convenience of each objective having its own annealing scheme (line 21), and hence unlike in [139], there is no need for normalizing the objectives in order to have a single temperature applied to the equation used in line 12.

In SA, an acceptance probability function (line 12) is used to check whether or not a newly generated solution is accepted [141]. As the temperature $\tau_{\acute{i}}$ tends to zero, the chance of accepting a worsened solution decreases and hence, the procedure acts as the "greedy algorithm" which only accepts "downhill " transitions. In Algorithm 5.2, a population is created for each generation by iteratively generating children (lines 4 to 5) through crossover and mutation operators, but only accepting those that dominate their parents (lines 9 to 11) or satisfy the acceptance criteria used by SA (lines 12 to 14). The notation $\succ$ is used to represent dominance. If the mutation operator is used to generate a child, $\mathcal{P}_{l,m}$ then only one of the input parents ($\mathcal{P}_{l-1,\acute{m}_1}$ or $\mathcal{P}_{l-1,\acute{m}_2}$) will be used, and consequently, only the used parent will be checked for dominance (i.e. in line 9). Due to the probabilistic nature of SA, when creating a new solution, the temperature is kept constant for a number of iterations (line 3) to try and obtain an acceptable solution. If after completing these iterations, an acceptable solution wasn't generated, then a parent is used as a solution (lines 15 to 17). Note that the chromosome representation needed for the NSGA-II can be based on the representation explained in Chapter 4.

5.3 Case Studies and Results

Two case studies are conducted using a real-world object (Fig. 5.3) and a complex simulated object (Fig. 5.4). The purpose is to test and compare the Stochastic-OMBP using different scenarios and conditions. Specifications of the AIRs used in the case studies are provided in Appendix A. Computation is carried out on a PC with 2.8GHz Intel Xeon E5-2680; however, since the code is single-thread then only one core of the CPU is used. The 'gamultiobj' function from the Matlab R2013a optimization toolbox, which is based on NSGA-II, was used. Parameters are set as default. Modifications were made to accommodate for the hybrid GA-SA approach.

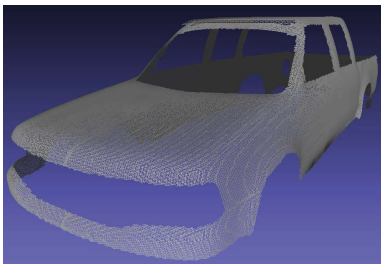
Main parameters used in the case studies are as follow: (i) targets are 0.04 m in radius and each has an approximately 30 % overlap with a neighboring target, (ii) discrete base placements are generated with 0.35 m spacing, (iii) the set-up time t_i^s in Eq. (5.7) is chosen to be a conservative value of 10 mins, (iv) δ in Eq. (5.9) is set to 1 m, and (v) in Eq. (5.2), the covariance $\mathbf{\Sigma}_i = \mathbf{\Sigma}$ for all i where $\mathbf{\Sigma}$ is a 2×2 identity matrix (I_2) multiplied by $\sigma^2 = 0.01\,\mathrm{m}^2$. The first joint of the AIRs which rotates about the z-axis is a full 360° revolute joint, and the base of the AIRs can't move vertically.

Note that at each base placement, the aim is to find the reachable targets rather than planning a trajectory for the AIR which requires a separate planner during the task execution. An independent lookup table (Appendix C) was used to generate AIR poses for each target representing the object.

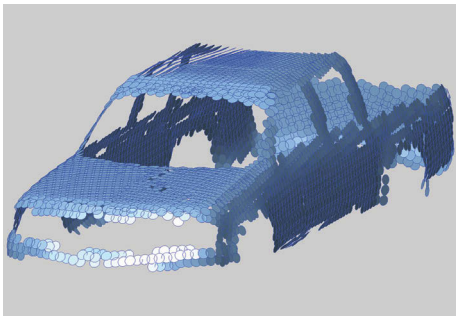
5.3.1 Case Study 1: Two AIRs Grit-blasting a Vehicle's Surfaces

There are 3270 targets representing the surfaces of the vehicle shown in Fig. 5.3c. The end-effector speed, v_i in Eq. (5.7) is set to 0.056 m/s for both AIRs. Figure 5.3d shows a solution chosen from the Pareto front. In the figure, 179 discrete base placements are shown, out of which 87 are the FBPs. The red filled circles (base placements) are poor performing in terms of surface coverage and the blue crossed circles have a high likelihood of collision with the object. The filled black circles are the chosen solution where the notation Ri:Bj represents the jth base placement of the ith AIR.

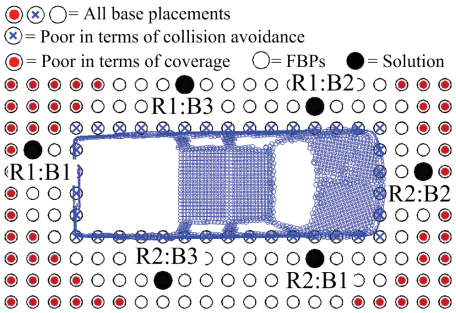
(a) Set-up of the experiment.



(b) Point cloud of the vehicle.



(c) Target representation.



(d) A solution selected from the Pareto front.

Fig. 5.3: An experiment using a vehicle where two AIRs are deployed to grit-blast the surfaces of a vehicle.

Based on the chosen solution, on average, 94.8 % of the reachable targets can be covered with a makespan of 3183 s. The difference in the completion time of the two AIRs is less than 3 % of the makespan, hence the AIRs will be expected to complete the task at approximately the same time. It was found that a solution with coverage greater than 99 % can be achieved using the model; however, this results in a significantly larger makespan (at least 12 % increase in the makespan). In the case when a slightly larger threshold for coverage is needed (e.g. 97 %), then at least one AIR will need to visit an additional base placement, which results in the makespan being significantly increased since the value of t_i^s is chosen to be conservative (10 mins).

5.3.1.1 Checking Solution Consistency

To ensure that the hybrid stochastic optimization algorithm used is robust and suitable for the problem under consideration, the optimization was repeated 10 times and for each run, the best solution from the Pareto front was selected. For each of the 10 solutions,

Table 5.1: Multi-objective hybrid GA-SA vs. NSGA-II.

	Coverage	SD	Makespan	SD
Hybrid GA-SA	92.82 %	0.0057	3167 (s)	30.79
NSGA-II	89.81 %	0.0189	3176 (s)	101.01

the sample average for each of the two objectives was calculated (i.e. based on Eqs. (5.4) and (5.8)) by considering 100 observations. Then, the average of the 10 solutions was calculated. The results are shown in Table 5.1. The low standard deviation (SD) in Table 5.1 shows that the optimization can generate consistent solutions.

5.3.1.2 Comparing Hybrid GA-SA with NSGA-II

To further demonstrate that the hybrid GA-SA algorithm is suitable, the optimization problem was solved using NSGA-II alone, i.e. without incorporating the Simulated Annealing (SA) algorithm. The same default settings were used for the 'gamultiobj' function of the Matlab optimization toolbox. Once again, the optimization was repeated 10 times and the average of the 10 solutions was calculated. 100 observations were considered in each solution. The results are shown in Table 5.1. It can be seen that the solutions obtained using the hybrid GA-SA algorithm are better than using NSGA-II alone. For both algorithms, on average, the optimization terminates at 105 GA iterations.

5.3.1.3 Testing for Various Levels of Uncertainties

To determine the extent to which uncertainties in base placements affect the results, a solution, Z^* is tested for different uncertainties, i.e. for different values of $\mathbf{\Sigma}$ ($\mathbf{\Sigma} = I_2\sigma^2$) in Eq. (5.2), as shown in Table 5.2. The solution, Z^* is obtained while assuming that uncertainties in base placement don't exist (i.e. not considering uncertainties in the model). Hence, Z^* has an exceptional performance (94 % coverage, and 3126 s makespan); however, this is only true for the condition where there are no uncertainties in base placements. The solution Z^* was then tested for different levels of uncertainties and as shown in Table 5.2, as the uncertainties in base placements of the AIRs increase, the result of Z^* worsens. Hence, by accounting for uncertainties as per the proposed Stochastic-OMBP method,

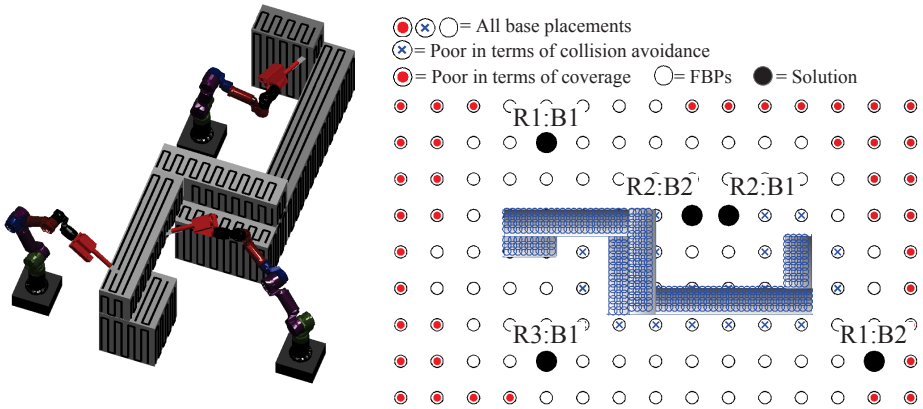
Table 5.2: Testing for different values of $\mathbf{\Sigma} = I_2\sigma^2$ (for the scenario where 2 AIRs are grit-blasting a vehicle).

		σ^2=0.01	σ^2=0.02	σ^2=0.03
For solution Z^*	Coverage (%)	91.25	88.60	86.16
	Makespan(s)	3131	3116	3108
Incorporating uncertainties	Coverage (%)	92.82	91.75	89.66
	Makespan(s)	3167	3107	3105

better solutions can be found, as shown in Table 5.2. Note that the solutions are calculated based on 100 observations.

5.3.2 Case Study 2: Three AIRs with Different Capabilities Grit-blasting a Complex Object

This case study aims to test the Stochastic-OMBP method for a complex structure shown in Fig. 5.4a, and for an increased number of AIRs (3 AIRs) that have different capabilities in terms of end-effector speed, i.e. $v_1 = 0.056\,\text{m/s}$, $v_2 = 0.05\,\text{m/s}$, and $v_3 = 0.04\,\text{m/s}$. In this case study, the acceptable threshold of coverage is set very high (99 %). The object can fit a minimal size bounding cuboid that is $3\,\text{m} \times 1\,\text{m} \times 0.8\,\text{m}$ in size. The three AIRs have the same workspace size and the same end-effector tool coverage. Thus, the discrete base placements shown in Fig. 5.4b are the same for all AIRs. 66 of the 135 discrete base placements are the FBPs.



(a) 3 AIRs grit-blasting a complex object.

(b) A solution selected from the Pareto front.

Fig. 5.4: A simulation where three AIRs are operating on a complex object.

The result is shown in Fig. 5.4b, which is based on a solution chosen from the Pareto front. AIR 3 is assigned only one base placement so as to minimize the overall completion time of the task since this AIR is the slowest AIR. On average, the completion time of AIRs 1 to 3 is 2060 s, 2060 s, and 1520 s, respectively. On average, the AIRs can cover 99.4 % of the 2279 targets representing the surfaces.

To check for consistency in results, the same procedure as in Case Study 1 (Section 5.3.1.1) is performed. The average of the 10 solutions is 99.3 % for coverage with a SD of 0.0032 %, and 2109 s for makespan with a SD of 94 s. On average, the optimization terminates at 104 GA iterations.

5.3.2.1 Testing for Various Levels of Uncertainties

The same procedure as in Case Study 1 (Section 5.3.1.3) is carried out for the scenario discussed in this case study. The results are shown in Table 5.3. Solution Z^* is optimal with 100 % coverage and a makespan of 2708 s; however, this is only true for the condition where it is assumed that there are no uncertainties in base placements. Solution Z^* becomes unacceptable in meeting the 99 % coverage threshold for all 3 levels of uncertainties considered in Table 5.3. By accounting for uncertainties, as per the proposed model, acceptable solutions are found for all 3 levels of uncertainties.

5.4 Discussion

The case studies showed that when there are uncertainties in base placements, then utilizing a mathematical model that takes uncertainties into account can provide solutions

Table 5.3: Testing for different values of $\mathbf{\Sigma} = I_2\sigma^2$ (for the scenario where 3 AIRs are grit-blasting a complex object).

		σ^2=0.01	σ^2=0.02	σ^2=0.03
For solution Z^*	Coverage (%)	98.90	97.94	97.16
	Makespan(s)	2150	2163	2179
Incorporating Uncertainties	Coverage (%)	99.26	99.49	99.23
	Makespan(s)	2109	2161	2325

that are better (more reliable during task execution) than the solutions provided by a mathematical model that doesn't take uncertainties into account.

An option to further improve the solution quality is to continue the optimization process during the task execution. That is, after obtaining the final solution from the Stochastic-OMBP method and during the task execution, the new information acquired from sensing the environment can be used to achieve a better solution. For example, after an AIR has moved to a base placement and if it localized itself more accurately, it can then update the ongoing optimization process with its relatively accurate base placement which can be used to obtain a better solution and to merge the current solution to a better solution at an appropriate time interval. Another potential improvement is to consider the sensor model to obtain a more representative distribution of the base placement uncertainties. However, the work in this thesis focuses on the higher level planning for complete coverage and hence, the effect of the selected localization algorithm or the sensor model on the performance of the AIRs is outside the scope of the thesis.

5.5 Conclusions

The Stochastic-OMBP method was presented with the aim of maximizing the coverage of objects' surfaces under uncertainties of the AIRs' base placements. The idea is to enable the AIRs to select base placements that minimize the effect of uncertainties on the performance of the AIRs, while optimizing the AIR team's objectives. A mathematical model has been developed to deal with the stochastic nature of the problem and to achieve complete coverage and minimal makespan. A stochastic optimization algorithm that considers hybrid multi-objective GA-SA was used to optimize coverage and makespan as per the developed mathematical model. The presented method was tested using both simulated and real-world objects and verified by means of several comparative studies using different scenarios and conditions.

Chapter 6

A Prey-Predator Behavior-Based Algorithm for Adaptive Real-Time Coverage Path Planning in Dynamic Environments

After performing base placement optimization (Chapters 4 and 5) and/or area partitioning and allocation (Chapter 3), each AIR will be allocated a surface area to cover. Each AIR must be able to generate appropriate end-effector tool paths on the allocated surfaces so as to achieve complete coverage. The AIR's end-effector tool, which is specific to the intended task (e.g. buffing pad or pressure cleaning nozzle), follows these paths. The task of generating end-effector tool paths on the surfaces for complete coverage is called Coverage Path Planning (CPP) [5]. Enabling an AIR to perform CPP in applications or environments where unexpected changes can occur, and possibly on objects with complex geometric shapes, is a challenging problem. A novel adaptive algorithm, which is based on the prey-predator behavior and the concept "risk of predation" [143], is presented in this chapter. The algorithm is termed Prey-Predator Coverage Path Planning (PPCPP).

As part of PPCPP, a total reward function is designed which incorporates rewards for minimizing the risk of predation, continuing motion in a straight direction, and covering boundary targets. For a given map of a surface, two parameters within the total reward function are optimized off-line. Then, the total reward function with the optimized parameters is used by an AIR in real-time to achieve complete coverage despite the changes in

the environment. Several case studies are presented[1] to verify the algorithm. Results show that PPCPP performs better than other adaptive CPP algorithms and that it enables an AIR to efficiently and effectively adapt its path to various changes.

The PPCPP algorithm enables a path to quickly adapt to unforeseen changes such as when stationary or dynamic obstacles (with initially unknown size and trajectory) are unexpectedly introduced into the environment or when an inaccurate base placement of an AIR (due to uncertainties in base placement) slightly changes the coverage area of interest. Furthermore, as will be shown, the algorithm can be applied to complex 3D surfaces, is computationally tractable for real-time implementation, and has only two parameters to tune for a given environment. Achieving optimal complete coverage, i.e. a path with minimal cost, is also taken into account in the algorithm.

The main results of this chapter were previously included in the following paper: Hassan and Liu [27].

The rest of the chapter is organized as follows. Section 6.1 provides a more detailed explanation of the problem. Section 6.2 presents the methodology which includes three sub-sections. Sub-section 6.2.1 briefly explains the prey-predator behavior. Sub-section 6.2.2 provides an overview of the algorithm followed by the mathematical model in Section 6.2.3. The mathematical model for optimizing the weighting factors is presented in Sub-section 6.2.4. Case studies are then presented in Section 6.3 to validate the algorithm. A brief discussion is presented in Section 6.4 and concluding remarks are stated in Section 6.5.

6.1 Problem Definition

Let *targets* be circular disks which represent a surface area. The definition of targets and the method to create targets were explained in detail in Section 1.3. The size and the density of the targets can be determined based on the intended application, the capabilities of the AIR, and the properties of the end-effector tool (e.g. the nozzle size in the spray

[1]Videos related to Case Studies 3, 5, 6 and 7 can be viewed at https://youtu.be/-zsoTqfM9IM, https://youtu.be/Wu1YuvCEBco, https://youtu.be/FZsEZ3Q8i10, and https://youtu.be/9veGRBIhZGQ, respectively.

painting AIR). After performing base placement optimization (Chapters 4 and 5) and/or area partitioning and allocation (Chapter 3), a set of targets is allocated to each AIR.

Given a set of targets on a surface that can be covered by an AIR at a base placement, the CPP problem is to plan a path such that the AIR's end-effector tool covers all the targets to achieve complete coverage of the surface. If unexpected stationary or dynamic obstacles appear in the environment or any changes to the coverage area suddenly occur, then the CPP problem is to re-plan the end-effector path of the AIR to cover the changed surfaces, i.e. to cover all *obstacle-free* targets.

The problem is challenging because: (i) the AIR is initially unaware of the obstacles (i.e. their size, location, and trajectory) or the changes in the environment, and (ii) the AIR is still expected to achieve complete coverage of the surface regardless of the unexpected changes that occur in the environment. This problem is further complicated when (i) the AIR, in addition to covering the whole surface amid unexpected changes, has to minimize the completion time or the path length; and (ii) the surface has a complex geometric shape.

When unexpected changes occur or a new obstacle is introduced to the environment, the AIR needs to make quick decisions on its next best direction of motion. The goal is therefore to develop a planner that is fast and adaptive while aiming to achieve an optimal path in real-time.

Each AIR has limited sensing range and can only detect obstacles within its sensing range. Sensing and predicting obstacles' trajectory [144] are outside the scope of the thesis. It is assumed that any obstacle can be detected and its trajectory can be predicted when the obstacle enters the sensing range of the AIR. It is also assumed that an appropriate motion/trajectory planning algorithm [10–17] for the AIR's manipulator joint motion is available and implemented. Hence, the work in this chapter only focuses on generating the complete coverage path on the surface and adapting it in real-time with respect to the changes in the environment. The AIR's end-effector is expected to be able to follow this path.

6.2 Methodology

6.2.1 Prey-Predator Behavior

The PPCPP algorithm is inspired by prey-predator behaviors and the concept "risk of predation"[143]. Various factors influence how animals assess fear in response to predation risk, including hunger, stress, amount of food, distance to a refuge, and predatory effects [145]. Although animals usually prefer foraging in habitats with a lower risk of predation, if food becomes abundant, then they lower their vigilance and select the region for foraging. This is because when food is abundant in an area, animals would wary that they may not find such abundance of food elsewhere and are more likely to stay for foraging [145]. However, when foraging "prey must select their optimal level of vigilance in response to their perceptions of a predator's whereabouts"[146]. Meaning that while foraging, animals continually assess the risk of predation. Using these concepts of risk evaluation, vigilance, and foraging in animals, an algorithm for coverage path planning is presented.

Fig. 6.1 shows a simple example which is used to explain the prey-predator behavior. A prey assumes a square shaped area to be a satisfying region for grazing due to abundance of food and its faraway distance from a predator. The start grazing point for the path of the prey is shown in the figure. The prey has a perception on the whereabouts of a potential ambushing predator which is outside of the area that is considered to be safe by the prey. Note that the predator may not be at the exact location where the prey has perceived it to be; however, the perception on the whereabouts of the predator is based

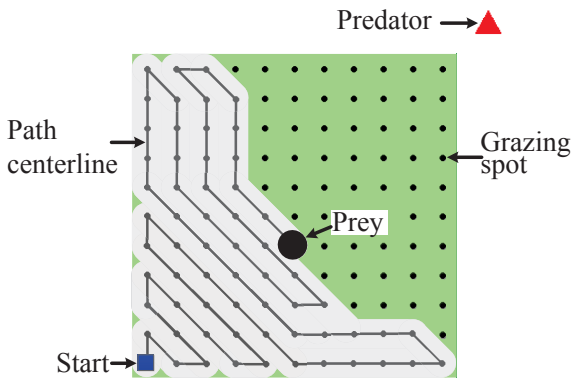


Fig. 6.1: A path by a prey due to grazing while avoiding a predator.

on the prey's experience, sensed noise, awareness of the habitat, etc. Hence, the prey maximizes its distance from the potential predator by first grazing in areas farthest away from the predator. However, since animals have to accomplish more than just avoiding predation, they need to continuously assess their current risk level of predation [147]. This means that while foraging, the prey consumes the food farthest away from the predator and gradually takes greater risks to move to locations closer to the predator. This behavior doesn't necessarily lead the prey to be caught by a predator since to avoid predation, animals have behavioral responses to predation risks such as increased vigilance [147]. In Fig. 6.1, the prey assumes that the food (e.g. grass) is of equal quality all over the field. Thus, as the prey moves to one of the neighboring unvisited spots, it will do so by assessing how far it will be from the predator to minimize its predation risk. Hence, in this paper, the foraging strategy of the prey is such that it aims to achieve the highest reward by moving to the next best spot that: (1) is not visited yet to maximize its chance of finding food, and (2) is farthest away from the predator. Maximizing this reward forms the basis of the concept behind the proposed approach. Therefore, it is understandable that since the prey maximizes its reward at each step, it will be forming a path as shown in Fig. 6.1, and will eventually end up in a grazing spot close to the predator.

6.2.2 The PPCPP Algorithm

The *prey* is considered as the coverage spot with a size equivalent to the coverage size of the AIR's end-effector tool. For example, for the spray painting or grit-blasting AIRs, the prey is the spray painting or the grit-blasting spot on the surface. The prey's location at step k is denoted as $\boldsymbol{o}_k$.

The *predator*, denoted as $\boldsymbol{\psi}$, is considered as a point outside the coverage area of the AIR and is inputted to the PPCPP. The predator is considered to be stationary (e.g. an ambushing predator). The location of the predator dictates the motion of the prey and how the path is built. The prey will always try to maximize its distance from the predator. Thus, the predator should be placed closest to a region of the coverage area (but outside the coverage area) where it is preferred for the coverage path to end, as shown in Fig. 6.1. The predator can also be placed inside one of the known obstacles.

Algorithm 6.1 PPCPP

Input: reachable targets O^r; start target $\boldsymbol{o}^s$; predator $\boldsymbol{\psi}$; weighting factors (ω^s, ω^b); No. of nearest neighbors $n^{N_{max}}$; neighborhood radius r; max. allocated time T

// Perform below in real-time:

Initialize: $k \leftarrow 1$; $O_k^u \leftarrow O^r$; $O_k^c \leftarrow \boldsymbol{o}^s$; $\boldsymbol{o}_k \leftarrow \boldsymbol{o}^s$; $O_k^{ob} \leftarrow \{\}$

while $O_k^u \neq \{\}$ **or** $t < T$ **do** // where $\{\}$ denotes empty set

// Scan the environment and update the map

$O_k^{ob} \leftarrow \text{Scan\&Update}(O^r, O_k^{ob})$

// Search for $n^{N_{max}}$ nearest neighbors (e.g. using k-d tree) and keep those that are within a distance r from $\boldsymbol{o}_k$

$N(\boldsymbol{o}_k) \leftarrow \text{NearestNeighbors}(O_k^u, \boldsymbol{o}_k, n^{N_{max}}, r)$

// Determine the obstacle-free neighbors of the prey

Initialize: $N^u(\boldsymbol{o}_k) \leftarrow \{\}$ // make set empty

for $i = 1$ to $\text{Size}\big(N(\boldsymbol{o}_k)\big)$ **do**

  if $\boldsymbol{o}_i \notin O_k^{ob}$ **then** // where $\boldsymbol{o}_i \in N(\boldsymbol{o}_k)$

    $N^u(\boldsymbol{o}_k) \leftarrow N^u(\boldsymbol{o}_k) \cup \boldsymbol{o}_i$ // add $\boldsymbol{o}_i$ to $N^u(\boldsymbol{o}_k)$

  end if

end for

// Determine the index of the neighbor with max. reward

Initialize: $R \leftarrow \{\}$

for $j = 1$ to $\text{Size}\big(N^u(\boldsymbol{o}_k)\big)$ **do**

  $N(\boldsymbol{o}_j) \leftarrow \text{NearestNeighbors}(O_k^u, \boldsymbol{o}_j, n^{N_{max}}, r)$ // where $\boldsymbol{o}_j \in N^u(\boldsymbol{o}_k)$

  $N^u(\boldsymbol{o}_j) \leftarrow \big\{\boldsymbol{o}_m \notin O_k^{ob}$ where $\boldsymbol{o}_m \in N(\boldsymbol{o}_j);\ \forall m : m = 1, 2, \ldots, \text{Size}\big(N(\boldsymbol{o}_j)\big)\big\}$

  // Calculate reward for $\boldsymbol{o}_j$ using Eqs. (6.1) to (6.4)

  $R_j \leftarrow \text{Reward}(\omega^s, \omega^b, \boldsymbol{o}_j, N^u(\boldsymbol{o}_j), \boldsymbol{o}_k, \boldsymbol{o}_{k-1}, n^{N_{max}}, \boldsymbol{\psi})$

end for

$j_k^* \leftarrow \arg\max_j(R_j)$

// Update states

$\boldsymbol{o}_{k+1} \leftarrow \boldsymbol{o}_{j_k^*}$ // prey moves to the best neighbor

$k \leftarrow k + 1$

$O_k^c \leftarrow O_{k-1}^c \cup \boldsymbol{o}_{j_k^*}$ // add $\boldsymbol{o}_{j_k^*}$ to covered targets

$O_k^u \leftarrow O_{k-1}^u \setminus \boldsymbol{o}_{j_k^*}$ // remove $\boldsymbol{o}_{j_k^*}$ from uncovered targets

$O_k^{ob} \leftarrow O_{k-1}^o$

end while

Algorithm 6.1 illustrates a pseudo-code of PPCPP. The next section provides a detailed mathematical model. In Fig. 6.1, the targets are the grazing or eating spots of the prey. As part of the input to the algorithm are the targets that are *reachable* by the AIR (obtained

using the methods in Chapters 4, 5 and/or 3), i.e. $O^r \subseteq O$ where O is a set containing all the targets that represent the surface. The starting location of the prey, $\boldsymbol{o}^s$, and the predator location, $\boldsymbol{\psi}$, are also inputted to the algorithm.

At each step, the prey is restricted to move to one of its neighboring targets for eating or grazing. The ith *neighbor* of $\boldsymbol{o}_k$, i.e. $\boldsymbol{o}_i \in N(\boldsymbol{o}_k)$, is defined as a neighboring target adjacent to $\boldsymbol{o}_k$. A target $\boldsymbol{o} \in O^r$ belongs to the neighboring set $N(\boldsymbol{o}_k)$ only if $\|\boldsymbol{o} - \boldsymbol{o}_k\| \leq r$. For a uniform decomposition of a surface, r is the distance to the diagonal neighbor (i.e. $r = \sqrt{a^2 + a^2}$ where a is the Euclidean distance from $\boldsymbol{o}_k$ to the nearest neighbor). For a uniform grid, this definition is equivalent to the 8-connectivity used in the image processing field.

To help the prey select the next best neighbor at each step, a total reward function, R, is designed. Using the R function at each step k ($k = 1, 2, \cdots, n^k$ where ideally n^k is equal to n^{O^r} which is the total number of targets in the set O^r) the AIR determines and moves to the neighbor with maximal reward. As will be explained in Section 6.2.3.1, the total reward function incorporates rewards for minimizing the risk of predation, continuing motion in a straight direction, and covering the boundary. Within the total reward function, there are two weighting factors, ω^s and ω^b (input to Algorithm 6.1) where ω^s is associated with the *smoothness* reward (i.e. the reward for the prey continuing motion in a straight direction) and ω^b is associated with the reward for covering the *boundary*. The values of the weighting factors govern the extent to which each reward is emphasized by the prey when deciding on the next movement, and will affect how the path is generated. Thus, the weighting factors need to be optimized for a given environment in order to achieve a path with minimal length. This optimization process for determining ω^s and ω^b needs to be carried out only once prior to real-time implementation of the algorithm and based on the initial knowledge of the environment (optimization procedure is explained in the next section). It is shown in the case studies (Section 6.3) that when the algorithm is used in real-time, the same optimized weighting factors are sufficient to manage changes in the environment effectively and to achieve complete coverage with a near-optimal path.

At each step, i.e. within each loop iteration (line 2 of Algorithm 6.1), the AIR determines and moves to the neighbor with maximal reward. The "while loop" in line 2 stops when

the prey eats all the food (when the set of uncovered targets, O^u_k, becomes empty) or when the current execution time, t, exceeds the maximum time, T, allocated to the coverage task.

At first, by scanning the environment (line 3), obstacles are detected and their trajectory and size are predicted. Then, the map of the environment is updated and the set O^{ob}_k which contains the targets that are predicted to be *occupied* by obstacles at step k is modified accordingly. That is, by performing the Scan&Update procedure in line 3, the aim is to remove the targets that are predicted to have become obstacle-free at step k from the set O^{ob}_k, and conversely, to add the targets that are predicted to have become occupied by obstacles to the set O^{ob}_k. Note that since the focus is on complete coverage, this procedure of predicting the obstacles and updating the map [144] is outside the scope of this thesis. As discussed in Section 6.1, it is assumed that the obstacles can be predicted at each step, and therefore, the set O^{ob}_k can be updated accordingly.

At each step k, the neighbors $N(\boldsymbol{o}_k) \in O^u_k$, of the prey, $\boldsymbol{o}_k$, are determined (line 4). Since a subset of the targets in O^u_k may become occupied by stationary or dynamic obstacles at step k, then the neighbors, $N(\boldsymbol{o}_k)$, need to be checked (lines 6 to 10). That is, each target $\boldsymbol{o}_i \in N(\boldsymbol{o}_k)$ is checked and if it is not occupied by any obstacle then it is added to the list of *uncovered* and *obstacle-free* neighbors, $N^u(\boldsymbol{o}_k)$ (lines 7 and 8).

From $N^u(\boldsymbol{o}_k)$, the index j^*_k (line 17), of the neighbor that results in maximum reward at step k is calculated (lines 11 to 17) and the corresponding best neighbor, $\boldsymbol{o}_{j^*_k}$, that the prey moves to is selected (line 18) where R_j in line 15 is the total reward for $\boldsymbol{o}_j \in N^u(\boldsymbol{o}_k)$. The states are then updated (lines 18 to 22), and the process is repeated for $k+1$th step (next loop). Note that as part of the input to the reward function for calculating the total reward associated with $\boldsymbol{o}_j$ are the uncovered and obstacle-free neighbors of $\boldsymbol{o}_j$, i.e. $N^u(\boldsymbol{o}_j)$ which are determined based on lines 13 and 14.

Using a k-d tree structure for finding the m-nearest neighbors (line 4) of a point (target) in a tree, the time complexity is $\mathcal{O}(m \log n)$ where n is the total number of points in the tree which is the same as n^{O^r} (number of targets in O^r). To check for a point in the tree (e.g. in line 7), the complexity is $\mathcal{O}(\log n)$. Thus, the time complexity of the first 'for-loop' (lines 6 to 10) for obtaining the obstacle-free neighbors is $\mathcal{O}(m \log n)$. For the

second 'for-loop' (lines 12 to 16), the time complexity is $\mathcal{O}(2m^2 \log n)$, since within the loop (i.e. for each j) the m-nearest neighbors for the target $\boldsymbol{o}_j$ need to be found (line 13) and each neighbor needs to be checked to determine whether or not it is in O_k^{ob} (line 14). To insert and delete a point from the k-d tree (lines 20 and 21), the time complexity is $\mathcal{O}(\log n)$. Therefore, the overall time complexity is $\mathcal{O}\big((2m^2 + m + 2) \log n\big)$, and the dominant part is $\mathcal{O}(m^2 \log n)$. Note that m (number of neighbors of any target) is small since CPP is concerned with the surface of an object. For uniform decomposition of the surface, the maximum number of neighbors is 8, as shown in Fig. 6.1. Thus, the running time for the AIR to make a decision for moving to the next best neighbor during the real-time implementation of PPCPP can be approximated as $\mathcal{O}(m^2 \log n)$. Note that the above analysis doesn't consider the time complexity of the algorithm used for sensing and prediction of obstacles (line 3). Depending on the application, scanning of the environment may not be needed for each step.

A *deadlock* is when the prey arrives at a target where all neighbors are already covered. In this case, the prey needs to repeat coverage of a certain number of targets in order to reach an uncovered target. If the prey arrives at a deadlock, PPCPP is temporarily stopped. Then, by switching to an existing point-to-point path planner that is suitable for the intended application, such as Dijkstra's algorithm, the shortest path from the deadlock target to the nearest uncovered target is found. PPCPP resumes when the prey reaches an uncovered target. The uncovered target to reach is an intermediate goal target and is a target closest to both the already covered targets and the prey. At each step, as the environment gets updated, the intermediate goal target can change or may no longer be needed. This is because at each step of the AIR's movement, the environment may change, e.g. an obstacle may move and the deadlock situation may no longer exist.

Suppose an environment is not subject to any changes and stationary or dynamic obstacles do not appear unexpectedly in the environment. Under such conditions, the PPCPP algorithm can achieve complete coverage of the surface because the main stopping criterion for the coverage task of the AIR is when all the targets in the set O_k^u are covered (line 2 of Algorithm 6.1). At each step k, the covered target by the AIR is added to the set O_k^c (line 20) and removed from the set O_k^u (line 18). This prevents the prey from getting stuck in revisiting targets since it is restricted in selecting a target from the set O_k^u only. The prey

is allowed to repeat coverage only when PPCPP is temporarily stopped as a result of a deadlock situation. PPCPP is resumed only after reaching an uncovered target. Hence, when the set O_k^u becomes empty, then the entire surface is covered and the coverage task is stopped. If the environment is subject to unexpected changes (e.g. due to dynamic obstacles), complete coverage of the surface is still possible; however, this is under the assumption that the obstacles can be detected and predicted accurately (line 3). The set O_k^u may not become empty due to a stationary obstacle unexpectedly occupying part of the surface that has not yet been covered by the AIR. In this circumstance, after covering all other obstacle-free targets, PPCPP will continue to attempt covering the occupied targets in the hope that they become unoccupied, until the current execution time, t, exceeds the maximum time, T, allocated to the coverage task.

6.2.3 Mathematical Model

In this section, the mathematical modeling related to the reward functions and optimization of the weighting factors is presented.

6.2.3.1 Reward Functions

Distance Reward: Reward for Moving Away from the Predator The farther away the prey is from the predator, the lower the risk of predation. Thus, at each step, the prey maximizes its reward by moving towards a neighbor that is uncovered and that has the farthest distance from the predator. The reward function for calculating the reward associated with moving to the jth neighbor $\boldsymbol{o}_j$ of the current prey target due to the distance from the predator is formulated as:

$$\mathrm{R}^d(\boldsymbol{o}_j) = \frac{\mathrm{D}(\boldsymbol{o}_j) - \mathrm{D}^{min}(\boldsymbol{o}_k)}{\mathrm{D}^{max}(\boldsymbol{o}_k) - \mathrm{D}^{min}(\boldsymbol{o}_k)} \tag{6.1}$$

where $\mathrm{D}(\boldsymbol{o}_j) = \|\boldsymbol{o}_j - \boldsymbol{\psi}\|$ gives the distance from the neighbor $\boldsymbol{o}_j$ to the predator $\boldsymbol{\psi}$, $\mathrm{D}^{max}(\boldsymbol{o}_k) = \max_j \|\boldsymbol{o}_j - \boldsymbol{\psi}\|$ gives the maximum distance from one of the neighbors of the current prey target to the predator, and similarly, $\mathrm{D}^{min}(\boldsymbol{o}_k) = \min_j \|\boldsymbol{o}_j - \boldsymbol{\psi}\|$ gives the minimum distance. Note that $\mathrm{D}^{max}(\boldsymbol{o}_k) - \mathrm{D}^{min}(\boldsymbol{o}_k)$ is therefore a constant for a

prey location. Based on Eq. (6.1), the prey obtains a maximum possible reward of 1 $\big(\mathrm{R}^d\big(\boldsymbol{o}_j\big) = 1\big)$ for moving to an uncovered neighbor that is farthest away from the predator, and conversely, it obtains zero reward $\big(\mathrm{R}^d\big(\boldsymbol{o}_j\big) = 0\big)$ for moving towards an uncovered neighbor closest to the predator. It gets a reward between 0 and 1 for moving to any of the other neighbors.

The foraging behavior of the prey due to this reward is such that at each step, the prey will move to an unvisited neighbor that is farthest away from the predator. At first, this behavior steers the prey towards the region farthest away from the predator. However, eventually the prey will have no choice but to gradually move closer and closer to the predator in search of new food. Thus, as was shown in Fig. 6.1, the aim of the prey is to search the whole area for food (obtain complete coverage) by gradually moving from the region farthest away from the predator to the region closest to the predator.

Angle Reward: Reward for Continuing Motion in a Straight Direction Although no specific evidence was found in the literature, it is reasonable to assume that a prey tends to move in a more regular pattern than the pattern shown in Fig. 6.1, i.e. it is more likely to move in a straight direction, make fewer turns, and try to cover the boundary. Thus, two additional prey-predator behaviors are considered, one of which is related to the prey continuing in the direction of its motion and only turning when it reaches a boundary or a deadlock. Having a path that has more straight lines (fewer turns) can be beneficial for certain AIRs, applications, or AIR motion planning algorithms. This reward function may be discarded for certain applications where having more straight lines doesn't necessarily improve the performance of the AIR.

The second reward function is formulated as follows:

$$\mathrm{R}^s\big(\boldsymbol{o}_j\big) = \frac{\angle \boldsymbol{o}_k^p \boldsymbol{o}_k \boldsymbol{o}_j}{180^\circ} \tag{6.2}$$

where $\mathrm{R}^s\big(\boldsymbol{o}_j\big) \in (0, 1]$ is the reward associated with the jth neighbor $\boldsymbol{o}_j$ of the current prey target $\boldsymbol{o}_k$ due to the angle $\angle \boldsymbol{o}_k^p \boldsymbol{o}_k \boldsymbol{o}_j \in (0^\circ, 180^\circ]$ which is the angle between the vectors $\big(\boldsymbol{o}_k^p - \boldsymbol{o}_k\big)$ and $\big(\boldsymbol{o}_k - \boldsymbol{o}_j\big)$, and $\boldsymbol{o}_k^p = \boldsymbol{o}_{k-1}$ is the preceding target that was covered by the prey at step $(k-1)$.

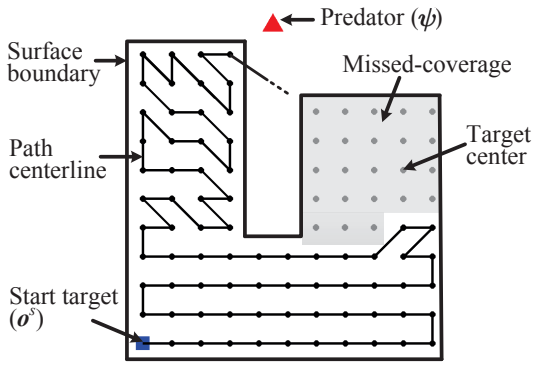


Fig. 6.2: The prey not covering certain regions due to close proximity to the predator.

Boundary Reward: Reward for Covering Boundary Targets Another prey-predator behavior is related to covering the boundary targets. Let the definition of boundary targets be the targets that represent the boundary of the surface as well as the targets that are on the boundary of the uncovered regions, i.e. the uncovered targets closest to the already covered region of the surface (targets closest to O_k^c). At each step, the prey will be given an extra reward for covering a boundary target, otherwise the prey may miss covering certain regions and it has to come back to finish covering these regions at a later stage which may cause a longer path. Figure 6.2 is an example of such a situation where the marked region is not covered (prior to reaching the target closest to the predator) since the targets in this region are closer to the predator than the targets the prey has decided to cover.

The third reward function is formulated as follows:

$$\mathrm{R}^b(\boldsymbol{o}_j) = \frac{n^{N_{max}} - \mathrm{N}^N(\boldsymbol{o}_j)}{n^{N_{max}}} \tag{6.3}$$

where $\mathrm{R}^b(\boldsymbol{o}_j) \in [0,1]$ is the reward associated with the jth neighbor $\boldsymbol{o}_j$ of the current prey target $\boldsymbol{o}_k$, and $\mathrm{N}^N(\boldsymbol{o}_j)$ calculates the number of uncovered neighbors of the target $\boldsymbol{o}_j$. $n^{N_{max}}$ is the maximum possible number of neighbors for a target and can be determined based on how the surface is decomposed. A value of 8 for $n^{N_{max}}$ is reasonable since CPP is concerned with the surface of an object. Hence, on a surface, a target can have a maximum of 8 neighbors if the aim has been to achieve a uniform decomposition of the surface. This reward function means that the smaller the number of uncovered neighbors for the target $\boldsymbol{o}_j$, the higher the reward.

6.2.3.2 Total Reward

Thus, the total reward for moving to an uncovered neighbor $\boldsymbol{o}_j$ is the sum of all the rewards stated previously, i.e.:

$$\mathrm{R}(\boldsymbol{o}_j) = \mathrm{R}^d(\boldsymbol{o}_j) + \omega^s\Big(\mathrm{R}^s(\boldsymbol{o}_j)\Big) + \omega^b\Big(\mathrm{R}^b(\boldsymbol{o}_j)\Big) \tag{6.4}$$

where ω^s and ω^b are the weighting factors associated with the smoothness and the boundary reward functions, respectively. Note that the geometric shape, the location and the distance between the start target and the predator, and the decomposition of the surface (arrangement of the targets) can be different each time the AIR is deployed for a new task. Hence, for a given environment, the weighting factors are first optimized; but only once before real-time implementation of the algorithm.

6.2.3.3 Maximum Reward at Step k

At step $k(k = 1, 2, \ldots, n^k)$, the index of the neighbor that gives the maximum reward can be found:

$$j_k^* = \arg\max_j \Big(\mathrm{R}\big(\boldsymbol{o}_j \in N^u(\boldsymbol{o}_k)\big)\Big) \tag{6.5}$$

where $\mathrm{R}(\boldsymbol{o}_j)$ is calculated based on Eq. (6.4). Thus at step k, the prey will move to the target $\boldsymbol{o}_{j_k^*} \in N^u(\boldsymbol{o}_k)$, the current prey target $\boldsymbol{o}_k$ will become $\boldsymbol{o}_{j_k^*}$, and the process is repeated until all targets are covered.

6.2.4 Mathematical Model for Optimizing the Weighting Factors

The prey, at each step, decides to move to the neighbor that gives the maximum total reward based on Eq. (6.5). However, the resulting complete path that the prey travels will only be optimal if the weighting factors within the total reward function (Eq. (6.4)) are optimized. Note that the optimization is performed off-line prior to the real-time implementation. The optimization iteratively changes the values of the design variables with the aim of improving the path length (cost). In each iteration, a complete path that covers the

entire surface is generated using the same process used for real-time implementation (Algorithm 6.1), however only utilizing the available knowledge of the environment. Note that optimizing the weighting factors may not be the only necessary condition for obtaining the globally optimal path; however, it is shown empirically (Section 6.3) that the optimized paths (using the optimized weighting factors) are near-optimal in most cases and that during the real-time implementation, the prey can adapt to changes in the environment while achieving a near-optimal path when compared to an optimized path.

The following is the mathematical model for optimizing the weighting factors.

6.2.4.1 Design Variables

The design variables, Z for the optimization problem are the weighing factors, i.e.:

$$Z = (\omega^s, \omega^b). \tag{6.6}$$

6.2.4.2 Objective Function

The cost of a path is defined with respect to its length. Thus, a path that covers all targets representing the surfaces of interest is considered optimal if its length is minimal. Minimizing the total length of the path has the added benefit of reducing the number of instances where the path crosses itself or ends up in a deadlock. This is because, for a complete coverage path to be minimal in length, the prey needs to prevent repeated coverage that is caused by the prey crossing its path or ending up in a deadlock. Recall that in a deadlock situation the prey needs to repeat coverage of a certain number of targets in order to reach the nearest uncovered target and then resume the coverage task.

The objective function for the optimization problem is:

$$\min_{Z} \mathrm{F}(Z) = \mathrm{L}^c(P^Z) \tag{6.7}$$

where P^Z is a path generated using the function π_1 which concatenates the targets in the proper sequence (iteratively based on Eq. (6.5)) with the current values of the design

variables taken into account:

$$\pi_1 : Z \mapsto P^Z = \left\{ \boldsymbol{o}^s, \boldsymbol{o}_{j_1^*}, \boldsymbol{o}_{j_2^*}, \ldots, \boldsymbol{o}_{j_{n^k}^*} \right\}, \tag{6.8}$$

hence a target in the generated path is determined by:

$$\pi_2 : j_k^* \mapsto \boldsymbol{o}_{j_k^*}. \tag{6.9}$$

where π_2 is the function that derives the corresponding target from the index j_k^* (in Eq. (6.5)).

Therefore, the length of the path

$$\mathrm{L}^c(P^Z) = \left\| \boldsymbol{o}^s - \boldsymbol{o}_{j_1^*} \right\| + \sum_{k=1}^{n^k-1} \left\| \boldsymbol{o}_{j_k^*} - \boldsymbol{o}_{j_{k+1}^*} \right\|, \tag{6.10}$$

and the aim is to obtain a path with minimal length through appropriate values of the design variables Z.

Achieving complete coverage by the prey is independent of the weighting factors. That is, as discussed previously, the algorithm stops when all the obstacle-free targets are covered. The values of the weighting factors provide a specific steering for the prey by dictating how much emphasis the prey should put on each reward when deciding on its next move. Hence, the value of the weighting factors will ultimately give the covering sequence of the targets and affect the optimality of the path in terms of length. However, the weighting factors are not part of the stopping criteria, nor do they impose a condition for the prey to stop the coverage task. Hence, the ability to achieve complete coverage under the assumptions discussed previously still holds true irrespective of the values of the weighting factors.

6.3 Case Studies and Results

Seven case studies, each with various scenarios or conditions, are used to validate the algorithm. In brief, the case studies include: Case Study 1 which verifies the reward

functions; Case Study 2 which consists of 8 different scenarios wherein each scenario a number of stationary obstacles with different shape and size are randomly placed in the coverage area of interest; Case Study 3 which compares the PPCPP algorithm with other adaptive algorithms; Case Study 4 which investigates the adaptability of the algorithm with respect to sudden change in the coverage area of a complex surface; Case study 5 which consists of 9 scenarios wherein each scenario a dynamic obstacle with different speed obstructs the end-effector path of an AIR; Case Study 6 which consists of 4 scenarios wherein each scenario multiple dynamic obstacles with different size and speed obstruct the end-effector path of an AIR; and Case Study 7 where a varying speed obstacle obstructs the end-effector path of an AIR operating on a complex surface.

The AIR has no prior knowledge of the stationary or dynamic obstacles that may become present in the environment and is unaware of their size, location, and trajectory. It only uses limited information (i.e. information it obtains at each step from its sensors) to update its local environment. At each step, the AIR only needs to know which of the prey's neighbors are obstacle-free and selects the neighbor that results in the maximum total reward. At each step, the reward is evaluated for a maximum of 8 neighbors. On average, for the case studies, it takes less than 1 millisecond for an AIR to compute the best neighbor that the prey should move to. This efficiency is necessary for the AIR to quickly decide on its next best move at each step amid dynamic obstacles. The optimization time for finding the optimal weighting factors is included in each case study. The cost of the returned solution (i.e. the path length) is also included. Note that as explained in Section 6.1, when the AIR senses its environment at each step, it is assumed that the AIR can accurately predict the size and trajectory of obstacles. However, since the AIR is not aware of the obstacles prior to the real-time implementation, it cannot rely on a preplanned path, and hence the planner needs to be efficient and adaptable to changes in the environment while still achieving complete coverage with a near-optimal path.

The code for testing the algorithm is written in Matlab. For a given environment, optimization for obtaining the optimal values of the weighting factors (as per the explanation in Section 6.2.4) is performed off-line using Genetic Algorithm (GA) through Matlab R2013a optimization toolbox. Other optimization algorithms can be utilized; comparison and selection of the most appropriate algorithm are outside the scope of this thesis. Parallel

computation is enabled in Matlab. The default settings of 'ga' from Matlab optimization toolbox were found to be appropriate for the case studies. As per the default settings, population size is 50 when the number of variables is less than or equal to 5, crossover fraction is 0.8, maximum generations is 100 * number of variables (2). All simulations were carried out using an Intel Core i5-2400 CPU @ 3.10 GHz and multi-threading using all four cores.

Note that in the subsequent figures, for a clearer visual demonstration of the results, only the center point of the targets and the centerline of the paths are shown.

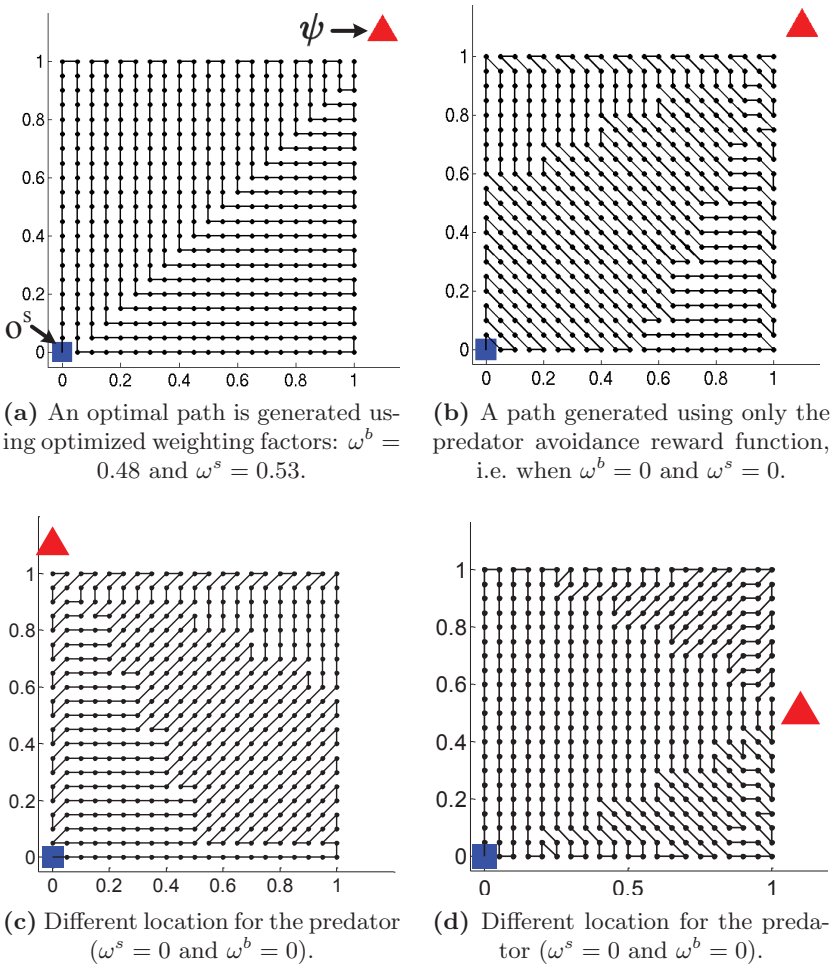
6.3.1 Case Study 1: Verifying Reward Functions

This case study is presented to demonstrate the benefits of having the smoothness and the boundary reward functions (i.e. the reward functions expressed in Eqs. (6.2) and (6.3)). In addition to the benefits mentioned in Sections 6.2.3.1 and 6.2.3.1, the two reward functions can jointly help with minimizing the total length of a path if optimized weighting factors are used. For the purpose of showing how different values of weighting factors affect the final path, a surface that is 1 m by 1 m in size is used. The surface is represented as 441 targets with a spacing of 0.05 m between each two non-diagonal neighbors.

PPCPP is performed on the surface considering two different pairs of values for the weighting factors: (i) optimized values, $\omega^s = 0.53$ and $\omega^b = 0.48$; and (ii) $\omega^s = \omega^b = 0$, meaning that the smoothness and the boundary reward functions are not used. The resulting paths due to these values are shown in Figs. 6.3a and 6.3b.

It can be seen in Fig. 6.3a that the PPCPP algorithm can return an optimal path if optimized weighting factors are used. The path in Fig. 6.3a is optimal in that it is the shortest possible path and doesn't cross itself. This optimal path is 22 m in length. Running the optimization to find optimal weighting factors took 46 s (average of 20 runs).

Longer paths may be generated when only considering the predator avoidance reward function (i.e. when $\omega^s = \omega^b = 0$), in this case 4.88 m longer than the optimized path which is 22 m, due to the diagonal line segments of the path. Thus, the two additional



(a) An optimal path is generated using optimized weighting factors: $\omega^b = 0.48$ and $\omega^s = 0.53$.

(b) A path generated using only the predator avoidance reward function, i.e. when $\omega^b = 0$ and $\omega^s = 0$.

(c) Different location for the predator ($\omega^s = 0$ and $\omega^b = 0$).

(d) Different location for the predator ($\omega^s = 0$ and $\omega^b = 0$).

Fig. 6.3: Paths generated on the surface using different weighting factors or a different location for the predator.

reward functions expressed in Eqs. 6.2 and 6.3 can help with minimizing the total length of a path and the smoothness reward function can reduce the number of turns.

Figures 6.3c and 6.3d are added to show that the prey covers the regions close to the predator near the end. Hence, the predator should be placed closest to a target where it is preferred for the coverage path to end.

To investigate convergence, optimization was repeated 20 times, and in all cases, an optimal solution was obtained. More interestingly, in each optimization run, an optimal solution was found within the first two GA iterations (generations) which in turn can be used to select a more appropriate number of GA generations and reduce the computation time. For the 20 optimization runs, on average, the optimization was terminated after 51 generations

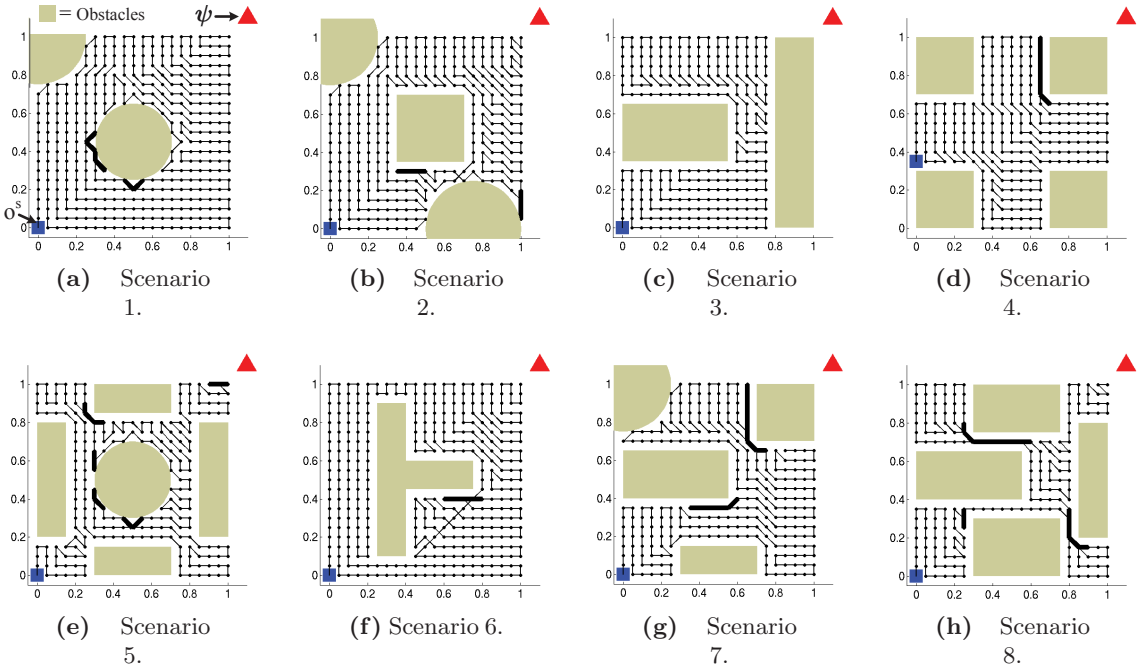
(note that maximum stall generations is 50). The quick convergence upon the solution can be as a result of the solution not being sensitive to small variations of the weighting factors. For the scenario under consideration, changing the value of ω^b from 0.4 to 0.9 while fixing the value of ω^s produces the same result. Conversely, changing the value of ω^s from 0.45 to 0.75 while fixing the value of ω^b produces the same result. It needs to be noted that the scenario presented in this case study is rather simple. Hence, convergence was quick and an optimal solution was found. For more complicated scenarios (e.g. for complex objects shown in the following case studies) optimization may take longer time to converge and an optimal solution may not be found.

6.3.2 Case Study 2: Adaptability Against Changes in the Environment

The purpose of this case study is to measure the adaptability of the PPCPP approach with respect to various changes in the environment. The PPCPP approach is also compared to optimal paths and paths generated using BA* algorithm which utilizes boustrophedon motion and A* search [148]. Eight scenarios are considered and in each scenario, a number of obstacles with different shape and size are randomly placed in the environment, as shown in Fig. 6.4. The same surface as the one presented in Case Study 1 (Section 6.3.1) is used.

For each scenario, the path that the prey travels in real-time (referred to as real-time PPCPP in Tables 6.1 and 6.2) is compared to an optimal path, an optimized path, and a boustrophedon path. The results are shown in Tables 6.1 and 6.2. The real-time PPCPP uses the optimized weighting factors, $\omega^s = 0.53$ and $\omega^b = 0.48$ obtained in Case Study 1.

Obtaining an optimal path for the sake of comparison is challenging. Even for a known environment without dynamic obstacles the problem reduces to the well know Traveling Salesman Problem (TSP) which is an NP-hard problem. To obtain the optimal path, a GA-based TSP is first used and then manual modifications are made with the aim of achieving the best representation of the optimal path. The comparison study is with respect to path length only; thus, an open-end TSP was implemented (i.e. the path is allowed to end at any target), and unlike PPCPP, smoothness of the path was not considered for the TSP. To check that the "optimal" path is truly optimal or at least near-optimal, it is compared to an ideal path that has no diagonal moves or overlaps, i.e. a path with a length equaling



(a) Scenario 1. **(b)** Scenario 2. **(c)** Scenario 3. **(d)** Scenario 4.

(e) Scenario 5. **(f)** Scenario 6. **(g)** Scenario 7. **(h)** Scenario 8.

Fig. 6.4: Eight different scenarios, and a path created for each scenario in real-time.

to [(number of targets − 1) × distance between two non-diagonal neighbors]. Note that such an ideal path may not exist. Taking the average of the results of the 8 scenarios, the percentage difference between the optimal paths and the ideal paths is 1.3 %.

The "optimized path" in Tables 6.1 and 6.2 is a path obtained by running the optimization process described in Section 6.2.4 while considering the changed environment assuming that the obstacles' location and size are known. The purpose of obtaining the optimized path is to determine how well the path generated in real-time using PPCPP compares to the optimized path. Taking the average of the results of the 8 scenarios, the percentage difference between the real-time PPCPP and the optimized paths is 1.6 %. The optimized paths using PPCPP are also compared to the optimal paths. Taking the average of the results of the 8 scenarios, the percentage difference between the optimized paths and the optimal paths is 5.2 %.

The paths generated in real-time using PPCPP are also compared to the on-line BA* algorithm [148] which utilizes boustrophedon motion. Taking the average of the results of the 8 scenarios, the percentage difference between the optimal paths and the boustrophedon paths is 7.35 %, whereas the percentage difference between the optimal paths and

Table 6.1: Result and comparison for each scenario: path lengths (m).

Scenario	Ideal	Optimal	Optimized	Real-time PPCPP	Boustro-phedon
1	18.25	18.49	19.12	19.37	18.83
2	15.20	15.33	16.21	16.64	16.21
3	12.55	12.57	12.91	13.01	13.27
4	12.20	12.24	12.91	13.08	12.86
5	10.75	10.90	11.73	12.06	12.62
6	17.20	17.20	17.80	18.07	17.98
7	12.85	12.91	14.11	14.19	13.60
8	9.30	9.83	10.19	10.36	11.30

Table 6.2: Result and comparison for each scenario: difference in lengths (m / %).

Scenario	Optimal to Ideal	Optimized to Optimal	Real-time PPCPP to Optimal	Real-time PPCPP to Optimized	Boustrophe-don to Optimal
1	0.24 / 1.32	0.63 / 3.41	0.88 / 4.76	0.25 / 1.31	0.34 / 1.84
2	0.13 / 0.86	0.88 / 5.74	1.31 / 8.55	0.43 / 2.65	0.88 / 5.72
3	0.02 / 0.16	0.34 / 2.70	0.44 / 3.50	0.10 / 0.77	0.70 / 5.54
4	0.04 / 0.33	0.67 / 5.47	0.84 / 6.86	0.17 / 1.32	0.62 / 5.10
5	0.15 / 1.40	0.83 / 7.61	1.16 / 10.64	0.33 / 2.81	1.72 / 15.75
6	0.00 / 0.00	0.60 / 3.49	0.87 / 5.06	0.27 / 1.52	0.78 / 4.56
7	0.06 / 0.47	1.20 / 9.30	1.28 / 9.91	0.08 / 0.57	0.69 / 5.34
8	0.53 / 5.70	0.36 / 3.66	0.53 / 5.39	0.17 / 1.67	1.47 / 14.96

the real-time PPCPP is 6.83 %. The difference between boustrophedon path and real-time PPCPP seems to be small; however, there are additional benefits to using PPCPP that is not shown to be present in algorithms that utilize boustrophedon motion. Boustrophedon-based approaches divide the surface into a number of regions; thus, it is more likely to have longer path lengths and overlap between paths due to transitions between regions. Using a boustrophedon approach, the path may not end close to a region of interest, whereas using PPCPP the path does end in the region closest to the predator as shown in Fig. 6.4. This behavior can be beneficial for some applications. Moreover, the path generated using PPCPP is more predictable since the prey tries to cover the region farthest away from the predator and gradually moves closer to the predator. Another advantage of PPCPP which is not shown in boustrophedon approaches is that it can handle dynamic obstacles and unexpected changes as will be shown in the following case studies.

Note that the AIR is initially unaware of the obstacles and their shape or size. At each step, the AIR scans a region around itself; it then determines which of the prey's neighbors

are obstacle-free and evaluates its next best move based on the total reward function. The AIR needs to evaluate the rewards for a maximum of 8 of the prey's neighbors at each step, hence computation time is efficient for real-time implementation (less than 1 millisecond). The prey does face deadlocks (dark black lines in Fig. 6.4) where it has to repeat coverage of certain targets so as to resume coverage of the uncovered region. These deadlocks cause longer paths; however, as shown in Fig. 6.4, the extra path lengths due to the deadlocks are relatively small.

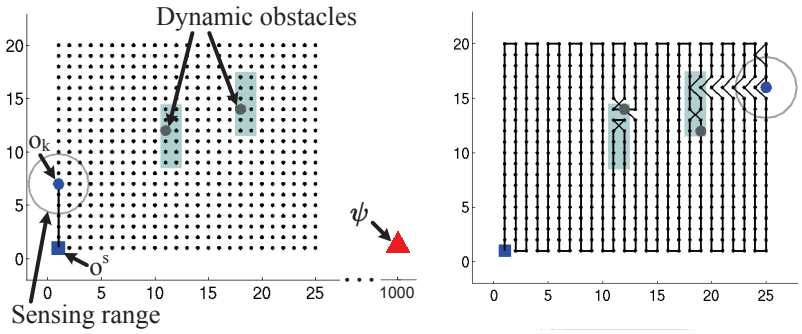
6.3.3 Case Study 3: Comparison with Other Adaptive Approaches

This case study is carried out to prove that the PPCPP algorithm can achieve better results when compared with a neural network-based approach, and an approach that incorporates rolling path planning and heuristic search to neural network [29]. As shown in Table 6.3, even though PPCPP only considers minimizing the path length, it also achieves better results in terms of the number of turns and the rate of repeated coverage for the scenario shown in Fig. 6.5 where two dynamic obstacles are present. In Fig. 6.5, the obstacle on the left continuously moves clockwise within the highlighted region, and the obstacle on the right moves counterclockwise. The scenario has been used by Qui et al. [29] to compare their combined approach to neural network only-based approach.

In neural network-based approaches, the surface is represented as a set of uniformly distributed neurons. The neural activity of the neurons that are in collision with obstacles is negative, whereas the neural activity of the neurons that are in the uncovered areas of the surface are positive. The positive neural activities propagate to the state space and globally attract the robot. The combined approach presented in [29] outperforms the neural network only-based approach mainly because, in a deadlock situation, it can find

Table 6.3: Comparison against other adaptive approaches.

	Path length	No. of turns	Repeated coverage (%)
Neural network	513	75	2.6
Approach presented in [29]	511	70	2.2
Proposed PPCPP algorithm	503	70	0.5



(a) A scenario with two dynamic obstacles.

(b) The path when obstacles' speed = 0.25 × robot's speed.

Fig. 6.5: A scenario where two dynamic obstacles continuously move within the highlighted rectangular regions, and an example path where the robot covers the whole surface.

the shortest path from the deadlock point to an intermediate goal point, and it only considers local information. The PPCPP algorithm outperforms both of the aforementioned approaches since it can immediately cover the free areas recently visited by the obstacles, whereas in the neural network-based approaches the robot can cover the areas recently visited by an obstacle only after all other areas are covered, because the neural activity of the neurons that have just been visited by an obstacle are lower than the uncovered neurons.

In the scenario shown in Fig. 6.5a, 500 targets (25 × 20) represent the surface. The environment is initially unknown to the robot and at each step, the robot can only scan a circular region around itself. It is assumed that the robot can accurately sense its local environment within a radius of 2.9 units as shown in Fig. 6.5a. Accurately sensing a circular region of 2.9 units will update 2 adjacent targets in each direction and it is just enough to update 2 diagonal targets in each direction since $\sqrt{(2^2 + 2^2)} = 2.83$ units. It is assumed that the robot senses and predicts the motion of the obstacle. The weighting factors are optimized only once using the first scan of the robot (i.e. when the prey is at its start location $\boldsymbol{o}^s$). Optimization takes less than 5 s using the 9 targets that fall within the sensing range of the robot at $\boldsymbol{o}^s$. The values for the optimized weighting factors are: $\omega^s = 0.52$ and $\omega^b = 0.20$, and the same weighting factors are used for the prey to cover the entire surface. Since the environment is initially unknown, the predator is placed at a very far away distance (1000 units) to the right of the prey's start point. Although this case study shows that the PPCPP algorithm may be able to handle unknown environments, a

more thorough study of this potential advantage needs to be carried out as future work for validation.

Note that in [29], the speed of the robot relative to the obstacles is not provided. Thus, for a fair comparison, the simulation is repeated three times, each time with a different speed, i.e. with obstacles' speed being quarter, half, and equal the speed of the robot. The result that is shown in Table 6.3 is based on the average of the solutions obtained from the three simulations. A supplementary video is available[2] at (https://youtu.be/-zsoTqfM9IM) which shows the motion of the prey and the path for all three simulations. Figure 6.5b shows the path corresponding to the scenario where obstacles' speed is quarter the speed of the robot. Also note that unlike in [29], the robot is not expected to return to its start point. Thus, all the results that are shown in Table 6.3 consider coverage path planning only, i.e. the prey stops when the whole surface is covered. Point to point path planning, such as heuristic search used in [29], can be added to PPCPP if the robot is required to return to its start point after it has finished covering the whole surface. In Table 6.3, path length, the number of turns, and the rate of repeated coverage is determined in the same way as in [29]. Note that neural network-based approaches are not shown to handle surfaces embedded in $\mathbb{R}^3$ and conditions where the decomposition of the surface is not uniform. However, the following case studies will demonstrate that PPCPP is capable of handling such conditions.

6.3.4 Case Study 4: Adaptability Against Changes in the Coverage Area of a Complex object

The purpose of this case study is to demonstrate that unexpected change in the reachable area can be managed using the adaptive behavior of the PPCPP. The change can be due to the inaccurate positioning of an AIR's base. It is shown that complete coverage with a near-optimal path can be achieved in real-time. For the applications and environments considered in this thesis (e.g. one-off autonomous grit-blasting), exploration and mapping by the AIR's sensors is necessary and results in a point cloud representation of the environment from which target representation is created. Although using sensor data may

[2] A video for Case Study 3 can be viewed at https://youtu.be/-zsoTqfM9IM.

cause non-uniform decomposition of the surface, PPCPP is shown to be practical for such conditions in the following case studies. Of course, if a CAD model is available, then uniform decomposition can be achieved which will help with obtaining a better path.

Fig. 6.6 shows a vehicle and an AIR to perform the one-off task of high-pressure cleaning on the vehicle. The AIR has calculated a set of base placements (e.g. using the method presented in Chapter 4 or 5) from which it will operate on the vehicle and clean all metallic surfaces of the vehicle. Assume that from one of these base placements the AIR can cover the 483 targets highlighted in Fig. 6.7a. However, since there are uncertainties associated with the AIR's base placement (e.g. due to localization and sensing errors), then the AIR becomes stationary at a position slightly different to the desired base placement. This mispositioning causes the AIR to cover the targets highlighted in Fig. 6.8a instead of the targets highlighted in Fig. 6.7a. All targets representing the vehicle are generated by considering a 0.0563 m distance between neighbors; however, since a point cloud is used to generate the targets, the distances between targets are not consistent, and the surface is not represented perfectly (i.e. decomposition of the surface is not uniform). Nevertheless, it is shown that PPCPP is capable of handling such inconsistency.

Note that manipulator motion planning with fixed or mobile base is not the focus of the thesis, but rather the focus is on planning the end-effector path for achieving complete coverage amid changes in the environment. However, as future work, it will be interesting to investigate and incorporate constraints related to the AIR's motion planning.

Prior to real-time implementation of the algorithm, optimization is performed using the 483 highlighted targets shown in Fig. 6.7a and the optimized weighting factors, $\omega^s = 0.45$

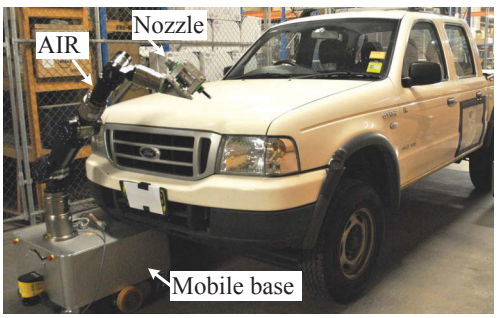
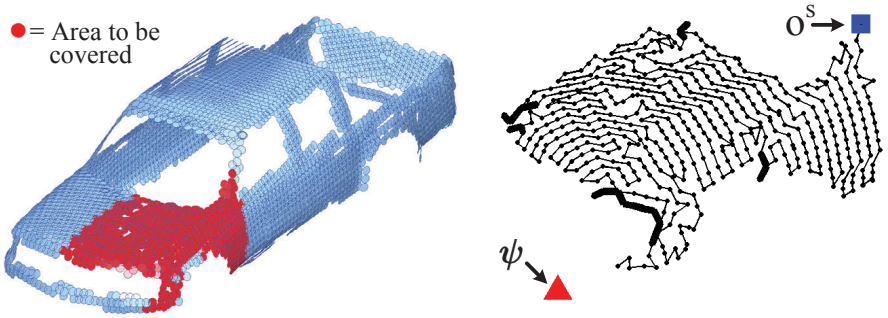


Fig. 6.6: An AIR is used to high-pressure clean a vehicle.



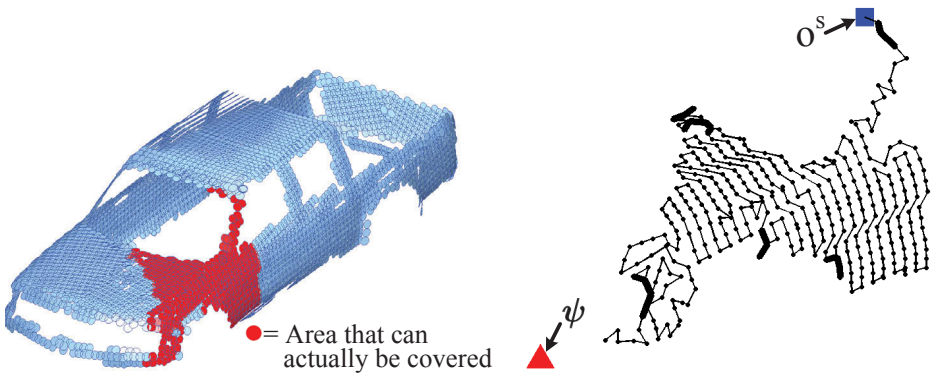
(a) Targets representing the vehicle and the areas expected to be covered.

(b) 3D view of the path corresponding to the areas highlighted in Fig. 6.7a.

Fig. 6.7: The areas expected to be covered by the AIR are shown. Optimization is performed to obtain appropriate weighting factors ($\omega^s = 0.45$ and $\omega^b = 0.96$) based on which the shown path is generated.

and $\omega^b = 0.96$ are obtained. Running the optimization to find optimal weighting factors took 78 s (average of 5 runs). The corresponding optimized path is shown in Fig. 6.7b. While generating the path, if the path ends up in a deadlock or if all neighbors are covered, the shortest path that goes through the already covered targets in order to reach the closest uncovered target needs to be found. These paths are shown as thicker black lines in Fig. 6.7b.

The AIR becomes stationary at a position slightly different to the desired base placement.



(a) Areas that can actually be covered from the current base placement of the AIR.

(b) 3D view of the path corresponding to the areas highlighted in Fig. 6.8a.

Fig. 6.8: The areas that can actually be covered by the AIR at its current base placement are shown, then the same weighting factors ($\omega^s = 0.45$ and $\omega^b = 0.96$) are used to generate the shown path.

Table 6.4: Results for 9 scenarios where in each scenario a single dynamic obstacle blocks the path of the AIR's end-effector.

Scenario / obstacle used	Radius of virtual sphere (m)	Speed ratio (prey : obstacle)	Path length (m)	Difference to optimized path (m / %)
1 / 1	0.11	2:1	22.79	1.27 / 5.9
2 / 2	0.11	2:1	22.23	0.71 / 3.3
3 / 3	0.11	2:1	22.96	1.44 / 6.7
4 / 1	0.17	1:1	22.19	0.67 / 3.1
5 / 2	0.17	1:1	21.86	0.34 / 1.6
6 / 3	0.17	1:1	22.43	0.91 / 4.2
7 / 1	0.23	2:3	23.92	2.40 / 11.2
8 / 2	0.23	2:3	23.55	2.03 / 9.4
9 / 3	0.23	2:3	24.76	3.24 / 15.1

At this inaccurate position, the AIR has to cover the updated set of targets (the 350 targets highlighted in Fig. 6.8a) using the same weighting factors, i.e. without repeating the optimization. The path of the AIR's end-effector (prey's path) to cover the updated set of targets is shown in Fig. 6.8b, which is 21.69 m in length. To compare prey's path with an optimized path, optimization is repeated for the changed environment, and an optimized path length of 21.52 m is obtained. This means that the optimized path is 0.17 m (0.8 %) shorter than the path that the prey traveled in real-time, which is insignificant. Running the optimization to find an optimal path (i.e. optimal weighting factors) for the updated set of targets took 57 s (average of 5 runs). During real-time implementation and at each step, it takes less than 1 millisecond to compute the best neighbor with maximal reward.

6.3.5 Case Study 5: Coverage in the Presence of a Dynamic Obstacle

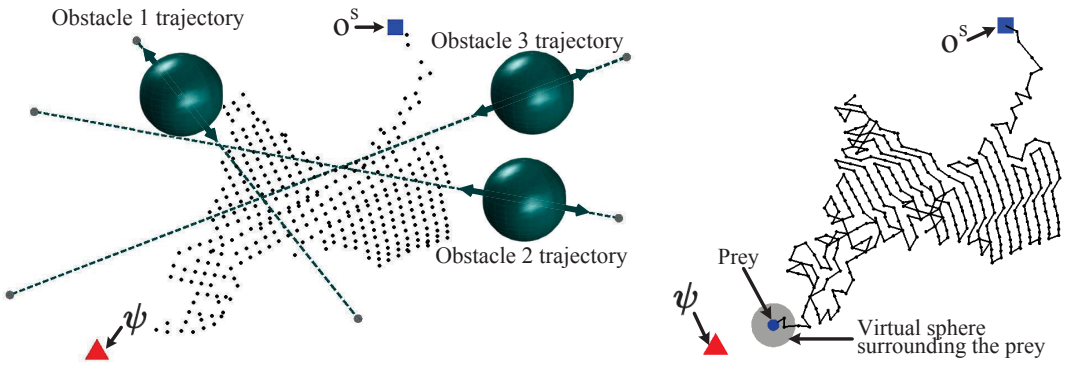
Nine scenarios are used in this case study to demonstrate that PPCPP can achieve complete coverage of a complex object while being adaptable to unexpected dynamic obstacles that can be faster or slower than the AIR's end-effector. The coverage areas of interest are the 350 highlighted targets that were shown in Fig. 6.8a. The same weighting factors, $\omega^s = 0.45$ and $\omega^b = 0.96$ as in Case Study 4 (Section 6.3.4) are used. Assume that an overhead crane moves an object (obstacle) across the workspace of the AIR while the AIR is in operation. A sphere can be used as an approximation of the volume that the obstacle occupies.

For each of the nine scenarios, a new obstacle with different trajectory or speed is used. The scenarios and the results are shown in Table 6.4. The results tend to show that when an obstacle is faster than the prey, the path length can be longer. This is because when the AIR detects an obstacle its top priority is to avoid collisions; as a result, repeated coverage can occur more often causing a longer path. In the table, the speed of the AIR corresponds to the prey's speed (i.e. end-effector coverage speed). The trajectory of each obstacle is shown in Fig. 6.9a. Obstacles continuously move back and forth along the shown trajectories. All obstacles are represented by a sphere having a 0.2 m radius. The end-effector of the AIR needs to move at an approximately constant speed for uniform coverage, i.e. the travel time between any two neighbors is considered to be constant.

As an example, the path corresponding to scenario 7 is shown in Fig. 6.9b. A supplementary video is available[3] at (https://youtu.be/Wu1YuvCEBco) which shows how the paths are generated for all nine scenarios.

Recall that the prey is the coverage spot of the end-effector, hence in the case study, the prey is the blasting or the cleaning spot. To prevent the prey from colliding with an obstacle, a virtual field (sphere in this case) is made to surround the prey (e.g. as shown in Fig. 6.9b). The virtual sphere can be made bigger if the obstacle is predicted to be faster than the AIR. Table 6.4 shows the size of the virtual sphere for each scenario. The size of the virtual sphere can also be made adaptive such that it becomes bigger when

[3]A video for Case Study 5 can be viewed at https://youtu.be/Wu1YuvCEBco.



(a) The trajectories of three spherical obstacles through the environment.

(b) An example path corresponding to scenario 7 where obstacle 1 is 1.5 times faster than AIR's end-effector.

Fig. 6.9: The trajectory of each obstacle is shown, and an example path is provided.

uncertainties in predicting the motion or speed of the obstacle are larger and vice versa. Since the obstacle can be faster than the AIR, it is possible for the obstacle to penetrate the virtual sphere surrounding the prey. In such a situation, the AIR's priority is to maximize the prey's distance from the obstacle until the obstacle is no longer inside the virtual sphere; then the task of complete coverage resumes. In the case where the prey is surrounded at a corner by an obstacle, then it is assumed that the AIR stops its operation (e.g. stops blasting or spray painting) and starts again at the nearest obstacle-free target that is not yet covered.

6.3.6 Case Study 6: Coverage in the Presence of Multiple Dynamic Obstacles having Different Speed and Size

The purpose of this case study is to show the effectiveness of PPCPP for more complicated conditions where multiple dynamic obstacles with different sizes and speeds block the path of the prey. As shown in Table 6.5, 4 scenarios are considered wherein each scenario 2 or 3 obstacles are used. Averaging the results of all 4 scenarios shown in the table, the difference between the generated path length and the optimal path length is 5.3 %. The same environment as in Fig. 6.9a is used; however, the obstacles' size and speed are changed as per the values shown in Table 6.6. The obstacles continuously move back and forth with the trajectories that were shown in Fig. 6.9a. The same weighting factors, $\omega^s = 0.45$ and $\omega^b = 0.96$ as in Case Study 4 (Section 6.3.4) are used.

As an example, the path corresponding to scenario 1 is shown in Fig. 6.10. Note that a virtual sphere of size 0.11 m surrounds the prey (end-effector point) and stops the prey

Table 6.5: Results for 4 scenarios where in each scenario multiple obstacles with different speeds and sizes block the path of an AIR's end-effector.

Scenario	Obstacles used	Path length (m)	Difference to optimized path (m / %)
1	1 & 2	22.72	1.20 / 5.6
2	1 & 3	22.12	0.60 / 2.8
3	2 & 3	22.17	0.65 / 3.0
4	1, 2 & 3	23.60	2.08 / 9.7

Table 6.6: Speed and size of each obstacle.

obstacle	Speed ratio (prey : obstacle)	Obstacles' radius size (m)
1	10:3	0.2
2	10:5	0.15
3	10:7	0.1

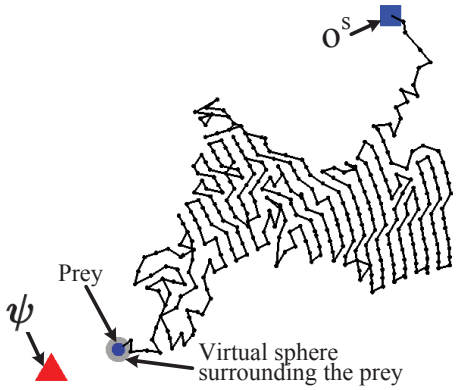


Fig. 6.10: An example path corresponding to scenario 1 of Table 6.5.

from colliding with the obstacles, as explained in Case Study 4 (Section 6.3.5). A supplementary video is available[4] at (https://youtu.be/FZsEZ3Q8i10) which shows how the path is generated for all four scenarios. In Table 6.6, the speed of the robot corresponds to the end-effector coverage speed (i.e. prey's speed).

6.3.7 Case Study 7: Coverage of a Complex Object in the Presence of a Varying Speed Obstacle

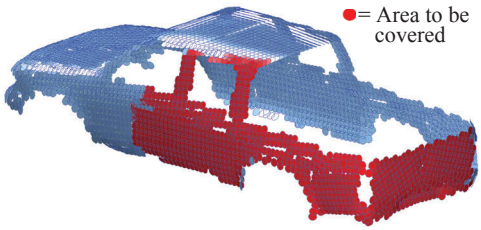
The purpose of this case study is to test PPCPP using another complex surface and for the condition where the obstacle has a varying speed. The speed of the obstacle can also exceed AIR's end-effector speed. The surface to be covered is highlighted in Fig. 6.11a where 886 targets represent the surface. Same as Case Study 4 presented in Section 6.3.4, target representation of the surface is not perfect (i.e. decomposition of the surface is not uniform). Nevertheless, it is shown that PPCPP is capable of handling this inconsistency.

The trajectory of the varying speed obstacle is shown in Fig. 6.11d. The obstacle moves through the points indicated on the trajectory as follows: point A to B, then to C, then

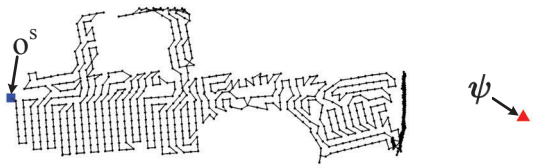
[4]A video for Case Study 6 can be viewed at https://youtu.be/FZsEZ3Q8i10.

back to B, and then back to A, with an end-effector to obstacle speed ratio of 10:20, 10:15, 10:5, and 10:10, respectively. The obstacle continuously repeats this trajectory. Since the obstacle can move faster than the AIR's end-effector (or the prey), a virtual sphere of size 0.28 m surrounds the prey and stops the obstacle from colliding with the prey, as explained in Case Study 5 presented in Section 6.3.5.

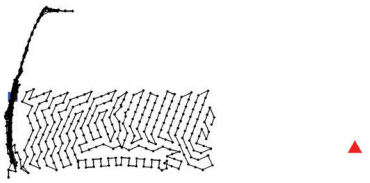
At first, optimization is performed off-line to obtain optimal weighting factors ($\omega^s = 0.63$ and $\omega^b = 1.55$) for the condition where it is assumed that there are no obstacles. The corresponding optimized path is shown in Figs. 6.11b and 6.11c. This path is 54.89 m in length.



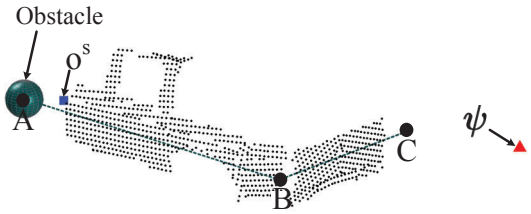
(a) Area to be covered.



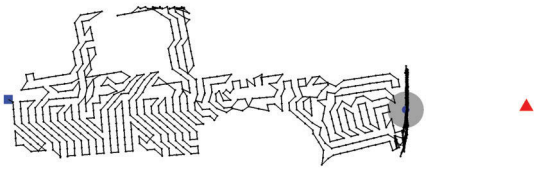
(b) Optimized path (view angle 1).



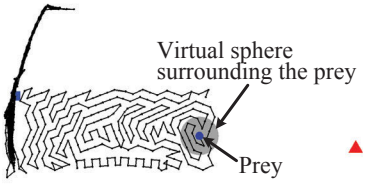
(c) Optimized path (view angle 2).



(d) Obstacle's trajectory.



(e) Path generated in real-time (view angle 1).



(f) Path generated in real-time (view angle 2).

Fig. 6.11: A scenario where an obstacle continuously moves through the area that needs to be covered with a varying speed.

During the real-time implementation, the AIR adapts its end-effector path with respect to the varying speed obstacle by determining, at each step, which of the prey's neighbors are obstacle-free and moving to the target that gives the maximum total reward (based on Eq. (6.5)). The corresponding path that the end-effector of the AIR travels is shown in Figs. 6.11e and 6.11f. A supplementary video is available[5] at (https://youtu.be/9veGRBIhZGQ) which shows how the path is generated. The path that is generated in real-time is 58.76 m (7.1 %) longer than the optimized path. The increase in the length of the path is mainly due to two reasons: (i) the prey moving away from the obstacle in many instances to avoid collisions, and (ii) collision avoidance causing deadlocks and repeated coverage. Nonetheless, this increase in the length of the path is acceptable for real-time coverage considering that the dynamic obstacle is present.

6.4 Discussion

The case studies verified that PPCPP is capable of achieving complete coverage and adapting to changes in an environment. Obtaining an optimal path in real-time where various changes can unexpectedly occur in the environment is computationally intractable. The goal has been to construct a planner that learns from the initial knowledge of the environment to optimize its parameters so that when applied in real-time, it can quickly adapt to the unexpected changes while still obtaining near-optimal coverage of the surfaces (when comparing to the optimized paths that are generated off-line). It was shown that the algorithm is efficient for real-time implementation because: (i) only limited information is needed at each step (i.e. information obtained at each step from the sensors) to update the local environment, and (ii) at each step, the AIR only needs to know which of the prey's neighbors are obstacle-free and selects the neighbor that results in maximum total reward.

Some additional concepts were briefly investigated in this chapter. For example, an unknown environment was used for Case Study 3 in Section 6.3.3; and virtual sphere was used in the case studies (Sections 6.3.4 to 6.3.7) to prevent the prey from colliding with an obstacle that is faster than the AIR's end-effector, but can also be used for handling

[5] A video for Case Study 7 can be viewed at https://youtu.be/9veGRBIhZGQ.

uncertainties in predicting obstacles' motion. A potential limitation of the PPCPP algorithm is related to the kinodynamic constraints of the AIR or the implemented motion planner with respect to the rate of response in pursuing the changing coverage path. Thus, these aspects need to be investigated when applying the PPCPP algorithm to a particular application and a particular AIR. As future work, these aspects can be studied.

6.5 Conclusions

A prey-predator behavior based algorithm to adaptive coverage path planning was presented in this chapter. The PPCPP algorithm is mainly designed for environments where unexpected changes can occur. Using many case studies, the algorithm was proven to enable an AIR to obtain complete coverage of target surfaces in real-time and with near-optimal path length amid various unforeseen changes in the environment. The PPCPP algorithm has only two parameters to optimize for a given environment before real-time implementation. Importantly, the algorithm was designed and proven to be adaptable to unexpected changes in the environment even if an object with complex geometric shape is to be operated on. It is also shown that the PPCPP algorithm is computationally tractable such that in the case of unexpected stationary or dynamic obstacles being present, the AIR can quickly respond based on the limited information it obtains from its sensors.

Chapter 7

Conclusions

The work in this thesis presented methodologies that were aimed at achieving optimal coverage by multiple AIRs. Examples of applications that require optimal coverage include grit-blasting, spray painting, and surface cleaning.

Below is a summary of the gaps in the literature that are addressed in this thesis:

- Most of the existing research work in complete coverage are focused on planar, approximately planar, or projectively planar (2.5D) environments. Examples include floor cleaning, harvesting, lawnmowing, and underwater surveying. However, AIRs are commonly deployed in environments where the objects have complex geometric shapes.
- Although there has been limited work on industrial robots operating in complex environments, their autonomy is limited. The majority of the industrial robots are still kept in factory settings and are purpose-built for a specific application. Thus, new methodologies are needed to improve the autonomous operation of industrial robots for coverage tasks. For example, taking uncertainties related to the the base placement of the AIRs into account will improve the solution. Furthermore, methodologies that directly use the point cloud information generated from sensing the environment are needed. This aspect is particularly helpful since, in real-life scenarios, exploration and mapping of unstructured environments are necessary.

- Very little attention has been given by researchers to devise methodologies for multiple AIRs to perform coverage task on complex objects. Mathematical models, with appropriate formulation of objectives and constraints, need to be devised so as to achieve optimal coverage by multiple AIRs.

- Unlike industrial robots deployed in controlled environments with prior knowledge of the objects, AIRs can be deployed in dynamic environments. Thus, dynamic obstacles and unexpected changes in the environment are to be taken into account.

More specifically, the problems addressed in this thesis are as follow: (i) partitioning and allocation of the surface areas of objects amongst the AIRs; (ii) determining optimal base placements for each AIR, with and without uncertainties taken into account; and (iii) adaptability of Coverage Path Planning (CPP) for dynamic environments and unexpected changes.

In solving these problems, it is vital for the AIRs to optimize the team's objectives, such as minimal makespan and minimal manipulator joints' torque, while accounting for relevant constraints, such as keeping a safe distance to obstacles.

7.1 Summary of Contributions

The main contributions of the work in this thesis are summarized below:

- A Voronoi partitioning based approach for simultaneous area partitioning and allocation of the overlapped areas (APA in Chapter 3): The developed mathematical model utilizes Voronoi partitioning to partition objects' surfaces, and multi-objective optimization to allocate the partitioned areas to the AIRs while optimizing AIR team's objectives. In addition to considering the objectives of minimizing the overall completion time and achieving complete coverage, torque and manipulability optimization of AIRs' manipulators are taken into account. The approach was proven to perform better than the pattern-based GA approach [62]. Case studies were presented to demonstrate the effectiveness of the approach for various objects including planar

objects, complex non-planar objects, multiple objects that are separate from each other, and real-world complex objects.

- An optimization-based method to multi-AIR base placement for complete coverage (OMBP in Chapter 4): The developed mathematical model addresses the problem of base placement, such that the AIRs collectively cover the entire object while optimizing team's objectives. The method accounted for AIRs with different capabilities and objects with complex geometric shapes. It included search space discretization and candidate base placements selection. It enabled AIRs to simultaneously determine: (i) an appropriate number of base placements, (ii) the location of the base placements, and (iii) the visiting sequence of the chosen base placements. Results from simulations and real-world experiments validated the method.

- A stochastic optimization-based method to multi-AIR base placement for complete coverage under uncertainties (Stochastic-OMBP in Chapter 5): The method aimed to maximize the coverage of objects' surfaces under uncertainties in the AIRs' base placements. The uncertainties are related to sensing and localization errors. The developed mathematical model handles the stochastic nature of the problem through Monte Carlo Simulations. The stochastic optimization was performed using hybrid multi-objective GA-SA which was found to be more efficient than using multi-objective GA alone. The presented method was tested using both simulated and real-world objects and verified through a number of comparative studies using different scenarios and conditions.

- A prey-predator behavior-based algorithm for adaptive and efficient real-time coverage path planning in dynamic environments (PPCPP in Chapter 6): The algorithm is inspired by the behaviors of animals in assessing fear with respect to predation risk and available resources. After each AIR is assigned surface areas at each base placement, CPP is to be performed to cover the allocated areas. The developed PPCPP algorithm enables an AIR to adapt its path in real-time so as to handle unexpected changes while still aiming to achieve minimal path length. Seven case studies, with various scenarios, were conducted to validate the algorithm. The results show that the difference between the length of the path generated in real-time

and the length of the optimal path generated off-line is small. The algorithm was compared to the neural network-based approaches, and it was found that the algorithm achieves better results in terms of path length, number of turns, and rate of repeated coverage.

7.2 Discussion on Limitations and Future Work

This thesis addressed a number of challenging problems that multiple AIRs are faced with when performing a complete coverage task. However, a number of limitations of the developed methods and algorithms should be addressed as part of future work.

The work in this thesis assumed that a reasonably accurate map of the objects is available. The given map is used to generate targets that represent the surfaces of objects. The targets are the input to the developed methodologies. The map of each object and the environment can be built using mapping methods if a CAD model of the object is not available, e.g. using the method explained in [7]. The accuracy of the map 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 [149]. However, studying these aspects was outside the scope of this thesis.

In the case studies, the base platform (which can be mobile) of the AIRs is kept static during the task execution, and when an AIR completes its task at a base placement, it moves to the next base placement using an existing and simple path planner. Keeping the base fixed while executing the task will help with the stability of the AIR and the accuracy in carrying out the task. Moving the base while executing the task does not necessarily improve productivity and may cause additional complications for the motion and trajectory planning. However, in some applications, it may be necessary to move the base while the AIR is executing the task. This would require additional parameters to be considered during the planning stage.

Once an AIR is allocated a surface area to cover, adaptive CPP is performed based on the developed PPCPP algorithm. Since the environment can be dynamic and unexpected changes can occur, then depending on the complexity of the changes that take place in the environment, the modified path from the PPCPP may not be optimal in terms of length. Aiming to obtain an optimal path in real-time where various changes can unexpectedly occur in the environment is computationally intractable. The goal is thus to have a path that can quickly adapt to the unexpected changes in an environment while still obtaining near-optimal coverage of the surfaces. Although a method for coping with obstacle-related uncertainties is included in the case studies (using a virtual field that surrounds the prey), the uncertainties associated with sensing and detecting the obstacles are to be further studied. A potential limitation of the PPCPP algorithm is related to the kinodynamic constraints of the robot or the implemented motion planner with respect to the speed at which adaptation to the changing coverage path is possible. Thus, these aspects need to be investigated when applying the proposed PPCPP algorithm to a particular application and a particular robot.

This research focused largely on optimality rather than computational efficiency. As was shown in the case studies, the developed methodologies were efficient in terms of computation time. However, there is room for improvements which can be investigated as future work.

Potential improvements in computational efficiency of the APA approach (Chapter 3) include: (i) using different optimization algorithms that can solve the problem more efficiently, (ii) restricting the seed points' location of the Voronoi graph by discretizing the search space, and (iii) representing the objects with fewer targets, meaning that less number of AIR poses need to be found.

The first and the third point outlined above for improving efficiency can also be applied to the base placement optimization methods presented in Chapters 4 and 5. Other potential improvements include: (i) using surface fitting to estimate the fitness values for the objective functions, and (ii) allowing the AIRs to start executing the task with a reasonable solution, and if a better solution is found at a later stage, then the current solution merges smoothly to the improved solution at an appropriate time interval.

The PPCPP algorithm was shown to be efficient for real-time implementation because it is a one-step planner and uses very limited information at each step. However, there are potential improvements or extensions that can be studied, including:

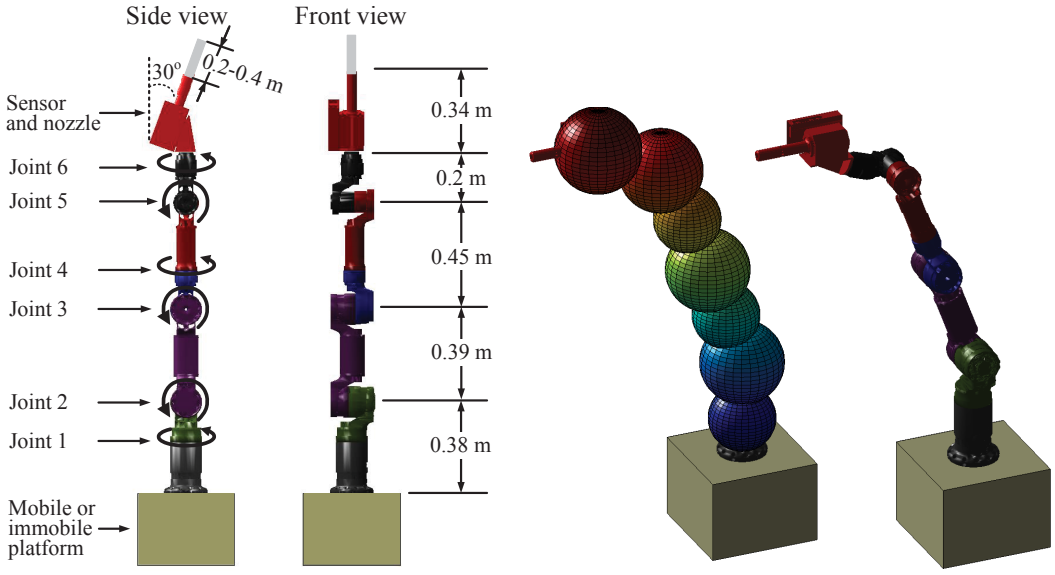
- Continuing optimizing the weighting factors while the AIRs are in operation, and updating the weighting factors at appropriate intervals to achieve a better path for a changed environment.
- Considering flexible location for the predator while optimizing the weighting factors, and switching to the new predator location if a better path is found.
- Considering multiple predators where both ambushing predators and dynamic predators (e.g. dynamic obstacles) can be considered for the prey to assess the risk of predation.
- Considering multi-step look-ahead for the AIR to have a higher chance of avoiding deadlocks.
- Integrating AIR's motion planning with PPCPP.

Other potential future works include: (i) determining the optimal number of AIRs needed for a given scenario and conducting a detailed investigation on the scalability of the method with respect to the number of AIRs, (ii) incorporating the sensor model to the coverage problem (e.g. to obtain a more representative distribution of the base placement uncertainties), and (iii) combining the developed methodologies into one global optimization model while maintaining the computational complexity to be tractable.

Appendix A

Specifications of the AIRs Used in the Case Studies

As shown in Fig. A.1a, the AIRs used in the case studies are comprised of an RGBD camera attached to a nozzle head, a 6 DOF Schunk industrial robot, and a mobile or an immobile platform. The base of the AIR can be mobile or can be immobile which will need manual positioning. Nonetheless, during the task execution (e.g. during the grit-blasting operation), the base is assumed to be fixed for better operation, stability, and accuracy



(a) Properties of the AIR. **(b)** Sphere representation of the AIR.

Fig. A.1: The AIR used in the case studies, it's properties and sphere representation.

when performing the task. After finishing part of a task at a base placement, the AIR then moves (or it is moved) to another base placement if necessary. Figure A.1a also shows the dimensions and properties of the AIR. The acceptable length of the grit-blasting or the spray painting stream from the tip of the nozzle to the target surface is considered to be 0.2 m to 0.4 m. The stream should also approach the surface with an angle greater than 10° and less than 50° relative to the surface normal. The end-effector speed and coverage size of the deployed AIRs can be different, and they are specified within the relevant case studies.

Fig. A.1b shows sphere representation of the AIR. The sphere representation is used for quick collision checking with the objects in the environment and to construct the lookup table (as described in Appendix C). Seven spheres are used to represent the manipulator and the nozzle of the AIR.

The manipulator of the AIR is constructed using 6 Schunk universal rotary actuators. The reference name and the related specifications of the actuators are provided in Table A.1. The joint limits considered for each actuator is also shown in Table A.1. The rotation limits (i.e. soft limits) are considered to be less than the Schunk actuators limits as a safety precaution.

Table A.1: Specification of the AIR's actuators.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Schunk actuator name	PRL120	PRL120	PRL100	PRL100	PRL80	PRL80
Weight (kg)	3.6	3.6	2.0	2.0	1.2	1.2
Nominal torque (N.m)	216	216	81.5	81.5	20.7	20.7
Soft joint limits (°)	±180	±110	±105	±175	±105	±175

Appendix B

Calculation of Joint Torque for a Given AIR Pose

In this appendix, the procedure for calculating the torque values for all joints of an AIR is explained. The torque values correspond to a single stationary AIR pose generated to reach a target. The specifications of the AIR analyzed henceforth are presented in Appendix A. The AIR consists of six revolute joints and no prismatic joints.

Let $\boldsymbol{F}^e$ define the forces f_x^e, f_y^e, and f_z^e along the x-, y-, and z-axes, and the moments m_x^e, m_y^e, and m_z^e about the x-, y-, and z-axes of the end-effector frame of an AIR; that is,

$$\boldsymbol{F}^e = [f_x^e \;\; f_y^e \;\; f_z^e \;\; m_x^e \;\; m_y^e \;\; m_z^e]^\mathsf{T}. \tag{B.1}$$

Let the end-effector frame be a 4×4 homogeneous transformation matrix where the first 3 vectors (columns) represent the orientation of the frame and the last vector represents the position of the frame [129, 150]. In the chapters, the end-effector frame describes the coverage spot (e.g. the blasting spot). For this appendix only, the end-effector frame describes the tip of an AIR's nozzle where, for example, a stream of grit or paint first exits the nozzle. The force on the end-effector is considered to be the reaction force caused by the high-pressure stream exiting the nozzle and is assumed to be constant. Since the reaction force is along the nozzle, then no moments are applied about the x-, y-, and z-axes of the end-effector frame. Thus, $\boldsymbol{F}^e$ becomes,

$$\boldsymbol{F}^e = [f_x^e \;\; f_y^e \;\; f_z^e \;\; 0 \;\; 0 \;\; 0]^\mathsf{T}. \tag{B.2}$$

Similarly, let the forces and moments at a frame j of the AIR's manipulator be:

$$\boldsymbol{F}_j = [f_x \;\; f_y \;\; f_z \;\; m_x \;\; m_y \;\; m_z]^\mathsf{T} \tag{B.3}$$

for $j \in 1, 2, \cdots, n^h$ where n^h is the number of joints (actuator) plus the one nozzle (a frame at the nozzle center). The force due to the weight of the links is relatively small and hence assumed negligible. The force $\boldsymbol{F}_j$ on the jth frame is due to the weight at the frame (e.g. the weight of an actuator or the nozzle). Thus, similar to Eq. (B.2), the moments m_x, m_y, and m_z are considered to be zero.

Let $\boldsymbol{q} = [\theta_1, \theta_2, \cdots, \theta_{n^J}]$ be an AIR manipulator pose defined using the angles of its n^J joints, and $\mathbf{T}^q(\boldsymbol{q})$ be a function that generates a matrix with the torque values of all n^J joints corresponding to an AIR pose $\boldsymbol{q}$. The torque values from $\mathbf{T}^q(\boldsymbol{q})$ are due to the end-effector reaction force $\boldsymbol{F}^e$, and gravitational forces on all frames $\boldsymbol{F}_j$ ($\forall j : j = 1, 2, \cdots, n^h$). That is,

$$\mathbf{T}^q(\boldsymbol{q}) = [\mathbf{J}^e(\boldsymbol{q})]^\mathsf{T} \boldsymbol{F}^e + \sum_{j=1}^{n^J} [\mathbf{J}_j(\boldsymbol{q})]^\mathsf{T} \boldsymbol{F}_j \tag{B.4}$$

where $\mathbf{J}^e(\boldsymbol{q})$ and $\mathbf{J}_j(\boldsymbol{q})$ give the Jacobian relative to the end-effector frame and the frame j, respectively. The Jacobian for an AIR pose $\boldsymbol{q}$ can be calculated based on the method in [129]. Note that the second term of Eq. (B.4) (i.e. after the plus sign) will only give the torque values of the relevant joints (i.e. the joints affected by the forces at the jth frame), hence other joints' torque values are given a value of zero to make the additions viable.

For the applications under consideration (e.g. spray painting or grit-blasting), the motion of the AIRs during task execution is slow. Thus, other factors such as angular, centripetal and Coriolis accelerations can be neglected when calculating torque. In other applications requiring high-speed movements, the dynamic forces should be taken into account.

Appendix C

Generation of AIR Pose Lookup Table

In order for an AIR, at a fixed base position, to cover the target o which represents part of a surface, the AIR needs to find a *feasible* pose q^f that reaches o with an appropriate end-effector position and orientation and without any collisions. There can be many feasible AIR poses that can reach and effectively cover o. However, it is desirable to select a feasible AIR pose that is better in terms of manipulability and torque. The purpose of finding feasible AIR poses for the targets that are inside the workspace boundary of an AIR is not to perform motion planning of the AIR's manipulator, but rather to check which targets are reachable and how good is the reachability in terms of manipulability measure and torque. There are a number of options for obtaining feasible AIR poses (i.e. performing inverse kinematics), which include: (i) analytical methods which are useful for a limited class of robotic manipulators, (ii) iterative or numeric methods (e.g. optimization-based) that aim to iteratively achieve better solutions, and (iii) using a lookup table which stores feasible AIR poses for a set of finely discretized end-effector points within the workspace of an AIR. The third option (using a lookup table) is implemented since it is computationally efficient and effective for the application. However, the accuracy is dependent on the size of the lookup table, i.e. the amount of information stored, and a larger lookup table will require a larger memory size to store and retrieve the data.

To generate the lookup table; first, the workspace of an AIR is decomposed into a large number of cube-shaped and equally-sized grids where each 3D grid is associated with many discrete AIR poses. The AIR poses associated with each 3D grid can be grouped based on

the similarity of their end-effector poses. The AIR poses associated with each group of a grid can then be sorted based on the manipulability measure, torque or other performance measures used in the AIR team's objectives. To find a feasible AIR pose for the target o, the AIR poses associated with the 3D grid that occupies o are checked in the sorted order, and the first collision-free pose is selected. Quadtrees, Octrees or similar hierarchical data structures [134] are used to perform fast searches so as to acquire the relevant data, e.g. to find the index of the 3D grid that occupies o.

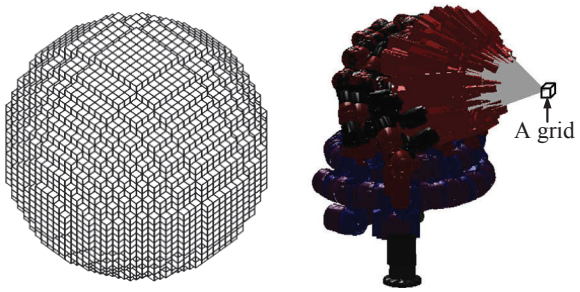
The construction of the lookup table is detailed in Algorithm C.1. First, the workspace of the AIR is decomposed into a large number of 3D grids (line 2 where G contains all grids). To do so, a sphere is used to approximate the workspace of an AIR. The radius of the sphere, r^s corresponds to the distance from AIR's base to the farthest point on the AIR's workspace boundary. The sphere is then decomposed into a large number of cube-shaped and equally-sized grids, as shown in Fig. C.1a. The side length of a grid, d^o can be decided based on the coverage size of the end-effector tool or the diameter of the targets used to represent the surface. The smaller the grids, the more accurate the results in finding feasible AIR poses; however, this is at the cost of utilizing more memory. Next, AIR poses are generated in line 3 where Q contains all AIR poses. AIR poses are generated by iteratively increasing the angle of each AIR joint by a small amount δ^a and then obtaining all possible permutations of the generated joint angles. The smaller the δ^a, the larger the number of AIR poses generated; however, more memory will be required

Algorithm C.1 Generate Lookup Table.

```
1: function GenerateLookupTable
2:     G ← Decompose(r^s, d^o)
3:     Q ← GeneratePoses(δ^a)
4:     for j = 1 to n^Q do
5:         E_j ← ForwardKinematics(q_j ∈ Q)
6:         W_j ← Manipulability(q_j)
7:         T_j^q ← Torque(q_j)
8:         G_j^o ← GridsOccupied(q_j)
9:         Data ← {q_j, E_j, W_j, T_j^q, G_j^o}
10:        Table ← Link2Grid(Table, G, Data)
11:    end for
12:    Table ← Sort(Table)
13:    return Table
14: end function
```

to store the lookup table. The function then loops through the permutations (i.e. AIR poses) in line 4. For each AIR pose (i.e. for each $q_j \in Q,\ j = 1, 2, \ldots, n^Q$), forward kinematics [150, 151] is performed (line 5) to obtain the end-effector frame $\boldsymbol{E}_j$ which is a 4 by 4 homogeneous transformation matrix representing the position and orientation of the end-effector. Note that the end-effector location represents the coverage spot on the surface. For each AIR pose, the manipulability measure (line 6) and the torques experienced by all joints (line 7) are calculated. The 3D grids $G_j^o \subseteq G$ that are occupied by the links, joints and nozzle of the jth generated AIR pose are also determined (line 8) by performing distance checking between the 3D grids and the centroid of the spheres which approximately represent the AIR. Predetermination of the 3D grids that an AIR pose occupies helps subsequent processes that perform collision checking with the objects in the environment. The data generated (line 9) based on q_j is then linked to a grid in G and is used to update the lookup table (i.e. $Table$ in line 10). To link q_j and its corresponding data to a grid, the end-effector position and orientation, i.e. the frame $\boldsymbol{E}_j$, is used to determine which grid contains the end-effector position and at what orientation the end-effector approaches the grid. Note that in applications such as grit-blasting and spray painting, the end-effector pose can follow a path within an acceptable range of angles with respect to the surface normal. Hence, the AIR poses associated with each grid need to be grouped based on the similarity of their end-effector pose. To do so, for each grid, several surface normals are considered. Then, each AIR pose that is associated with a grid is linked to a surface normal based on the orientation of the end-effector relative to each surface normal of a grid. For example, Fig. C.1b shows a set of AIR poses that are grouped together since the end-effectors approach a grid within a predefined angle relative to a surface normal. Thus, $Table$ contains information about each AIR pose q_j as well as the grid and the group that q_j belongs to. The AIR poses associated with each group of each grid are sorted (line 12) based on the quality of each pose in terms of manipulability measure, torque or the weighted average of the two parameters.

The lookup table is generated only once. To find a feasible AIR pose from the lookup table to reach a target, the following procedure is performed: (i) transformation of relevant data (e.g. AIR's base and targets) to an origin where the lookup table was originated, (ii) determining the grid that is occupied by the target using distance checking, (iii) retrieving



(a) Grid representation. **(b)** A group of AIR poses.

Fig. C.1: The workspace of an AIR is approximated as a sphere and is represented as many cube-shaped and equally-sized grids. Each grid is associated with many groups where each group contains many AIR poses that can reach the grid within a predefined angle relative to a surface normal.

the appropriate group of AIR poses associated with the selected grid by checking the difference between the target's normal and the normals associated with the grid, and (iv) looping through the AIR poses within the selected group in the sorted order and obtaining the first collision-free AIR pose in the group which is better than the subsequent candidate AIR poses in terms of the calculated manipulability measure and/or torque. To ensure that an AIR pose is collision-free, the grids in G_j^o corresponding to the AIR pose are checked to determine whether or not they occupy any physical obstacle surface from the surrounding environment. Note that Quadtrees, Octrees or similar data structures [134] are used to perform fast distance queries, e.g. in step 1.

Table C.1 summarizes the main properties of the lookup table used in the case studies. If a grid is not associated with any AIR poses, then the grid is considered to be unreachable by the AIR (i.e. it is outside the workspace of the AIR). The larger the number of AIR poses in a group associated with a grid, then the greater the dexterity of the AIR when reaching the grid with the particular end-effector orientation.

Table C.1: Main properties of the lookup table used in the case studies.

Description	Value
Data structure used	Octree
Number of levels for Octree	6
Number of 3D grids used to represent the workspace of the AIR	262144
Number of groups of AIR poses associated with each 3D grid	67
Number of AIR poses associated with each group	0 to 300

Bibliography

[1] N. Papakostas, G. Michalos, S. Makris, D. Zouzias, and G. Chryssolouris. Industrial applications with cooperating robots for the flexible assembly. *International Journal of Computer Integrated Manufacturing*, 24(7):650–660, 2011.

[2] D. Liu, G. Dissanayake, P. B. Manamperi, P. A. Brooks, G. Fang, G. Paul, S. Webb, N. Kirchner, P. Chotiprayanakul, N. M. Kwok, et al. A robotic system for steel bridge maintenance: research challenges and system design. In *Australasian Conference on Robotics and Automation*, pages 3–5, Dec 2008.

[3] G. Paul, S. Webb, D. Liu, and G. Dissanayake. A robotic system for steel bridge maintenance: field testing. In *Australasian Conference on Robotics and Automation*, pages 1–8, 2010.

[4] M. Hvilshoj, S. Bogh, O. Skov Nielsen, and O. Madsen. Autonomous industrial mobile manipulation (AIMM): past, present and future. *Industrial Robot: An International Journal*, 39(2):120–135, 2012.

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

[6] C. S. Mathew and D. K. Varghese. Robotic bridge maintenance system: A comprehensive study. *International Journal of Engineering Technology Science and Research*, 3(3):123–131, 2016.

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

[8] G. Paul, D. Liu, N. Kirchner, and G. Dissanayake. An effective exploration approach to simultaneous mapping and surface materialtype identification of complex three-dimensional environments. *Journal of Field Robotics*, 26(11/12):915–933, 2009.

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

[10] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni. Trajectory planning in robotics. *Mathematics in Computer Science*, 6(3):269–279, 2012.

[11] J. C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[12] P. Chotiprayanakul, D. Liu, D. Wang, and G. Dissanayake. A 3-dimensional force field method for robot collision avoidance in complex environments. In *Proceedings of the 24th International Symposium on Automation and Robotics in Construction*, pages 19–21, Sep 2007.

[13] W. K. To, G. Paul, N. M. Kwok, and D. Liu. An efficient trajectory planning approach for autonomous robots in complex bridge environments. *International Journal of Computer Aided Engineering and Technology*, 1(2):185–208, 2009.

[14] A. Shukla, E. Singla, P. Wahi, and B. Dasgupta. A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces. *Robotics and Autonomous Systems*, 61(2):209–220, 2013.

[15] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[16] N. E. Du Toit and J. W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2012.

[17] R. Saravanan, S. Ramabalan, C. Balamurugan, and A. Subash. Evolutionary trajectory planning for an industrial robot. *International Journal of Automation and Computing*, 7(2):190–198, 2010.

[18] G. Peters and G. Paul. Maintaining an old icon with a new technology. *Journal of Protective Coatings & Linings*, 32(8):22, 2015.

[19] G. Paul, N. Kwok, and D. Liu. A novel surface segmentation approach for robotic manipulator-based maintenance operation planning. *Automation in Construction*, 29:136–147, 2013.

[20] G. Paul, D. K. Liu, and N. Kirchner. *An Algorithm for Surface Growing from Laser Scan Generated Point Clouds*, pages 481–491. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[21] M. Hassan, D. Liu, S. Huang, and G. Dissanayake. Task oriented area partitioning and allocation for optimal operation of multiple industrial robots in unstructured environments. In *13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1184–1189, Dec 2014. doi: 10.1109/ICARCV.2014.7064473.

[22] M. Hassan and D. Liu. Simultaneous area partitioning and allocation for complete coverage by multiple autonomous industrial robots. In *Autonomous Robots*, pages 1–20, 2017. doi: 10.1007/s10514-017-9631-3.

[23] M. Hassan and D. Liu. An approach to base placement and area partitioning for complete surface coverage by multiple autonomous industrial robots. In *4th International Doctoral Symposium on Mechanical Engineering at Hokkaido University (IDSHU)*, pages 89–94, Nov 2015.

[24] M. Hassan, D. Liu, G. Paul, and S. Huang. An approach to base placement for effective collaboration of multiple autonomous industrial robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3286–3291, May 2015. doi: 10.1109/ICRA.2015.7139652.

[25] M. Hassan, D. Liu, and G. Paul. Collaboration of multiple autonomous industrial robots through optimal base placements. In *Journal of Intelligent and Robotic Systems*, Oct 2017. doi: 10.1007/s10846-017-0647-x.

[26] M. Hassan, D. Liu, and G. Paul. Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2978–2984, Oct 2016. doi: 10.1109/IROS.2016.7759461.

[27] M. Hassan and D. Liu. A prey-predator behavior based approach to adaptive coverage path planning. In *Transactions on Automation Science and Engineering (T-ASE), Conditionally accepted - Feb 2018*.

[28] M. Hassan, D. Liu, and D. Xu. A two-stage approach to optimization of multi-robot collaborative fiber placement. In *Journal of Intelligent and Robotic Systems (Submitted Jan 2018)*.

[29] X. Qiu, J. Song, X. Zhang, and S. Liu. A complete coverage path planning method for mobile robot in uncertain environments. In *The Sixth World Congress on Intelligent Control and Automation*, volume 2, pages 8892–8896, 2006.

[30] H. Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001.

[31] J. H. Lee, J. S. Choi, B. H. Lee, and K. W. Lee. Complete coverage path planning for cleaning task using multiple robots. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3618–3622, Oct 2009.

[32] H. Lee and A. Banerjee. Intelligent scheduling and motion control for household vacuum cleaning robot system using simulation based optimization. In *Winter Simulation Conference*, pages 1163–1171, Dec 2015.

[33] M. Taix, P. Soueres, H. Frayssinet, and L. Cordesses. Path planning for complete coverage with agricultural machines. In S. Yuta, H. Asama, E. Prassler, T. Tsubouchi, and S. Thrun, editors, *Field and Service Robotics*, volume 24 of *Springer Tracts in Advanced Robotics*, pages 549–558. Springer Berlin Heidelberg, 2006.

[34] J. Conesa-Munoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro. Efficient distribution of a fleet of heterogeneous vehicles in agriculture: A practical approach to multi-path planning. In *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 56–61, Apr 2015.

[35] I. A. Hameed. Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *Journal of Intelligent & Robotic Systems*, 74(3):965–983, 2014.

[36] Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

[37] S. P. Fekete, S. Friedrichs, A. Kröller, and C. Schmidt. Facets for art gallery problems. *Algorithmica*, 73(2):411–440, 2015.

[38] B. Englot and F. S. Hover. Sampling-based coverage path planning for inspection of complex structures. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2012.

[39] J. A. Broderick, D. M. Tilbury, and E. M. Atkins. Optimal coverage trajectories for a UGV with tradeoffs for energy and time. *Autonomous Robots*, 36(3):257–271, 2014.

[40] K. Jeddisaravi, R. J. Alitappeh, L. C. A. Pimenta, and F. G. Guimarães. Multi-objective approach for robot motion planning in search tasks. *Applied Intelligence*, 45(2):305–321, 2016.

[41] K. M. Hasan, A. Al-Nahid, and K. J. Reza. Path planning algorithm development for autonomous vacuum cleaner robots. In *International Conference on Informatics, Electronics Vision*, pages 1–6, May 2014.

[42] E. M. Arkin, S. P. Fekete, and J. S.B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(12):25 – 50, 2000.

[43] M. Burger, M. Huiskamp, and T. Keviczky. Complete field coverage as a multi-vehicle routing problem. *IFAC Proceedings Volumes*, 46(18):97–102, 2013.

[44] X. Yu, T. A. Roppel, and J. Y. Hung. An optimization approach for planning robotic field coverage. In *Annual Conference of the IEEE Industrial Electronics Society*, pages 4032–4039, Nov 2015.

[45] T. Oksanen and A. Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[46] D. Batsaikhan, A. Janchiv, and S.G. Lee. *Sensor-Based Incremental Boustrophedon Decomposition for Coverage Path Planning of a Mobile Robot*, pages 621–628. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[47] G. P. Strimel and M. M. Veloso. Coverage planning with finite resources. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2950–2956, Sept 2014.

[48] S. C. Wong and B. A. MacDonald. A topological coverage algorithm for mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1685–1690, Oct 2003.

[49] L. H. Nam, L. Huang, X. J. Li, and J. F. Xu. An approach for coverage path planning for UAVs. In *International Workshop on Advanced Motion Control*, pages 411–416, Apr 2016.

[50] K. R. Guruprasad and T. D. Ranjitha. ST-CTC: A spanning tree-based competitive and truly complete coverage algorithm for mobile robots. In *Proceedings of Advances In Robotics*, AIR '15, pages 43:1–43:6, New York, NY, USA, 2015. ACM.

[51] M. Yan, D. Zhu, and S. X. Yang. A novel 3-D bio-inspired neural network model for the path planning of an AUV in underwater environments. *Intelligent Automation & Soft Computing*, 19(4):555–566, 2013.

[52] C. Luo, S. X. Yang, H. Mo, and X. Li. Safety aware robot coverage motion planning with virtual-obstacle-based navigation. In *IEEE International Conference on Information and Automation*, pages 2110–2115, Aug 2015.

[53] S. Jeon, M. Jang, D. Lee, Y. J. Cho, and J. Lee. Multiple robots task allocation for cleaning a large public space. In *SAI Intelligent Systems Conference (IntelliSys)*, pages 315–319, Nov 2015.

[54] D. Ball, P. Ross, A. English, T. Patten, B. Upcroft, R. Fitch, S. Sukkarieh, G. Wyeth, and P. Corke. *Robotics for Sustainable Broad-Acre Agriculture*, pages 439–453. Springer International Publishing, Cham, 2015.

[55] D. Michel and K. McIsaac. New path planning scheme for complete coverage of mapped areas by single and multiple robots. In *2012 IEEE International Conference on Mechatronics and Automation*, pages 1233–1240, Aug 2012.

[56] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu. A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints. *IEEE Transactions on Cybernetics*, 44(3):305–314, March 2014.

[57] S. Bhattacharya, R. Ghrist, and V. Kumar. *Multi-robot Coverage and Exploration in Non-Euclidean Metric Spaces*, pages 245–262. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[58] A. Borkar, A. Sinha, L. Vachhani, and H. Arya. Collision-free trajectory planning on lissajous curves for repeated multi-agent coverage and target detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1417–1422, Oct 2016.

[59] X. Cao and D. Zhu. Multi-AUV task assignment and path planning with ocean current based on biological inspired self-organizing map and velocity synthesis algorithm. *Intelligent Automation & Soft Computing*, 0(0):1–9, 0.

[60] B. Ranjbar-Sahraei, G. Weiss, and A. Nakisaee. A multi-robot coverage approach based on stigmergic communication. In I. Timm and C. Guttmann, editors, *Multiagent System Technologies*, volume 7598 of *Lecture Notes in Computer Science*, pages 126–138. Springer Berlin Heidelberg, 2012.

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

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

[63] A. Janchiv, D. Batsaikhan, B. Kim, W. Lee, and S.G. Lee. 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, 2013.

[64] M. K. Gunady, W. Gomaa, and I. Takeuchi. Aggregate reinforcement learning for multi-agent territory division: The hide-and-seek game. *Engineering Applications of Artificial Intelligence*, 34(0):122–136, 2014.

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

[66] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero. Distributed approach for coverage and patrolling missions with a team of heterogeneous aerial robots under communication constraints. *International Journal of Advanced Robotic Systems*, 10, 2013.

[67] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart. Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots. *Autonomous Robots*, 40(6):1059–1078, 2016.

[68] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero. A decentralized algorithm for area surveillance missions using a team of aerial robots with different sensing capabilities. In *IEEE International Conference on Robotics and Automation*, pages 4735–4740, May 2014.

[69] J. J. Acevedo, B. C. Arrue, J. M. Diaz-Bañez, I. Ventura, I. Maza, and A. Ollero. One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots. *Journal of Intelligent & Robotic Systems*, 74 (1):269–285, 2014.

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

[71] X. Cao, D. Zhu, and S. X. Yang. Multi-AUV target search based on bioinspired neurodynamics model in 3-D underwater environments. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–11, 2015.

[72] Michael R. and F. Xi. Coverage based tool-path planning for automated polishing using contact mechanics theory. *Journal of Manufacturing Systems*, 30(3):144 – 153, 2011.

[73] P. N. Atkar, D. C. Conner, A. Greenfield, H. Choset, and A. A. Rizzi. *Uniform Coverage of Simple Surfaces Embedded in R3 for Auto-Body Painting*, pages 27–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[74] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.

[75] T. Danner and L. E. Kavraki. Randomized planning for short inspection paths. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 971–976, Apr 2000.

[76] I. Maza and A. Ollero. *Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms*, pages 221–230. Springer Japan, Tokyo, 2007.

[77] C. Di Franco and G. Buttazzo. Energy-aware coverage path planning of UAVs. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 111–117, Apr 2015.

[78] L. Paull, S. Saeedi, M. Seto, and H. Li. Sensor-driven online coverage planning for autonomous underwater vehicles. *IEEE/ASME Transactions on Mechatronics*, 18 (6):1827–1838, Dec 2013.

[79] E. Galceran and M. Carreras. Efficient seabed coverage path planning for ASVs and AUVs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 88–93, Oct 2012.

[80] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao. Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. *Journal of Field Robotics*, 32(7):952–983, 2015.

[81] T. S. Lee, J. S. Choi, J. H. Lee, and B. H. Lee. 3-D terrain covering and map building algorithm for an AUV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4420–4425, Oct 2009.

[82] T.S. Lee and B. H. Lee. A new hybrid terrain coverage method for underwater robotic exploration. *Journal of Marine Science and Technology*, 19(1):75–89, 2014.

[83] A. Xu, C. Viriyasuthee, and I. Rekleitis. Efficient complete coverage of a known arbitrary environment with applications to aerial operations. *Autonomous Robots*, 36(4):365–381, 2014.

[84] H. Chen, T. Fuhlbrigge, and X. Li. Automated industrial robot path planning for spray painting process: A review. In *IEEE International Conference on Automation Science and Engineering*, pages 522–527, Aug 2008.

[85] W. Sheng, H. Chen, N. Xi, and Y. Chen. Tool path planning for compound surfaces in spray forming processes. *IEEE Transactions on Automation Science and Engineering*, 2(3):240–249, Jul 2005.

[86] P. N. Atkar, D. C. Conner, A. Greenfield, H. Choset, and A. A. Rizzi. Hierarchical segmentation of piecewise pseudoextruded surfaces for uniform coverage. *IEEE Transactions on Automation Science and Engineering*, 6(1):107–120, Jan 2009.

[87] X. Maldague I. Mantegh P. Olivieri, L. Birglen. Coverage path planning for eddy current inspection on complex aeronautical parts. *Robotics and Computer-Integrated Manufacturing*, 30(3):305 – 314, 2014.

[88] Z. Bo, F. Fang, S. Zhenhua, M. Zhengda, and D. Xianzhong. Fast and templatable path planning of spray painting robots for regular surfaces. In *34th Chinese Control Conference*, pages 5925–5930, July 2015.

[89] T. R. Ren, N. M. Kwok, D. K. Liu, and S. D. Huang. Path planning for a robotic arm sand-blasting system. In *International Conference on Information and Automation*, pages 1067–1072. IEEE, 2008.

[90] P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar. Exact cellular decomposition of closed orientable surfaces embedded in R3. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 699–704, 2001.

[91] L. Li, D. Xu, X. Wang, and M. Tan. A survey on path planning algorithms in robotic fibre placement. In *27th Chinese Control and Decision Conference (CCDC)*, pages 4704–4709, May 2015.

[92] P. Debout, H. Chanal, and E. Duc. Tool path smoothing of a redundant machine: Application to automated fiber placement. *Computer-Aided Design*, 43(2):122–132, 2011.

[93] B. Shirinzadeh, G. Cassidy, D. Oetomo, G. Alici, and M. H. Ang Jr. Trajectory generation for open-contoured structures in robotic fibre placement. *Robotics and Computer-Integrated Manufacturing*, 23(4):380–394, 2007.

[94] L. Yan, Z. C. Chen, Y. Shi, and R. Mo. An accurate approach to roller path generation for robotic fibre placement of free-form surface composites. *Robotics and Computer-Integrated Manufacturing*, 30(3):277–286, 2014.

[95] X. Zhang, W. F. Xie, and S. V. Hoa. Modeling and workspace analysis of collaborative advanced fiber placement machine. In *International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2014.

[96] N. Agmon, N. Hazon, and G. A. Kaminka. The giving tree: constructing trees for efficient offline and online multi-robot coverage. *Annals of Mathematics and Artificial Intelligence*, 52(2):143–168, 2008.

[97] H. H. Viet, S. Choi, and T. Chung. An online complete coverage approach for a team of robots in unknown environments. In *13th International Conference on Control, Automation and Systems*, pages 929–934, Oct 2013.

[98] H. H. Viet, V. H. Dang, M. N. U. Laskar, and T. Chung. BA*: an online complete coverage algorithm for cleaning robots. *Applied Intelligence*, 39(2):217–235, 2013.

[99] Y. H. Choi, T. K. Lee, S. H. Baek, and S. Y. Oh. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5788–5793, Oct 2009.

[100] I. Shnaps and E. Rimon. Online coverage of planar environments by a battery powered autonomous mobile robot. *IEEE Transactions on Automation Science and Engineering*, 13(2):425–436, Apr 2016.

[101] I. Shnaps and E. Rimon. Online coverage by a tethered autonomous mobile robot in planar unknown environments. *IEEE Transactions on Robotics*, 30(4):966–974, Aug 2014.

[102] J. Hess, M. Beinhofer, and W. Burgard. A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots. In *IEEE International Conference on Robotics and Automation*, pages 5600–5605, May 2014.

[103] T. Bretl and S. Hutchinson. Robust coverage by a mobile robot of a planar workspace. In *IEEE International Conference on Robotics and Automation*, pages 4582–4587, May 2013.

[104] C. Das, A. Becker, and T. Bretl. Probably approximately correct coverage for robots with uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1160–1166, Sept 2011.

[105] L. Paull, M. Seto, and H. Li. Area coverage planning that accounts for pose uncertainty with an AUV seabed surveying application. In *IEEE International Conference on Robotics and Automation*, pages 6592–6599, May 2014.

[106] M. F. Aly, A. T. Abbas, and S. M. Megahed. Robot workspace estimation and base placement optimisation techniques for the conversion of conventional work cells into autonomous flexible manufacturing systems. *International Journal of Computer Integrated Manufacturing*, 23(12):1133–1148, 2010.

[107] J. Yang, W. Yu, J. Kim, and K. Abdel-Malek. On the placement of open-loop robotic manipulators for reachability. *Mechanism and Machine Theory*, 44(4):671 – 684, 2009.

[108] P. Sotiropoulos, N. Aspragathos, and F. Andritsos. Optimum docking of an unmanned underwater vehicle for high dexterity manipulation. *IAENG International Journal of Computer Science*, 38(1):48–56, 2011.

[109] P. Sotiropoulos, N. Tosi, F. Andritsos, and F. Geffard. Optimal docking pose and tactile hook-localisation strategy for AUV intervention: The DIFIS deployment case. *Ocean Engineering*, 46(0):33 – 45, 2012.

[110] S. Mitsi, K. D. Bouzakis, D. Sagris, and G. Mansour. Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm. *Robotics and Computer-Integrated Manufacturing*, 24(1):50–59, 2008.

[111] G. C. Vosniakos and E. Matsas. Improving feasibility of robotic milling through robot placement optimisation. *Robotics and Computer-Integrated Manufacturing*, 26 (5):517–525, 2010.

[112] G. Boschetti, R. Rosa, and A. Trevisani. Optimal robot positioning using task-dependent and direction-selective performance indexes: General definitions and application to a parallel robot. *Robotics and Computer-Integrated Manufacturing*, 29 (2):431–443, 2013.

[113] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent advances in graph partitioning. *Preprint*, 2013.

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

[115] M. Ghosh, N. M. Amato, Y. Lu, and J.M. Lien. Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2):494–504, 2013.

[116] O. V. Kaick, N. Fish, Y. Kleiman, S. Asafi, and D. Cohen-OR. Shape segmentation by approximate convexity analysis. *ACM Trans. Graph.*, 34(1):4:1–4:11, Dec 2014.

[117] S. Asafi, A. Goren, and D. Cohen-Or. Weak convex decomposition by lines-of-sight. *Computer Graphics Forum*, 32(5):23–31, 2013.

[118] J.M. Lien and N. M. Amato. Approximate convex decomposition of polygons. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 17–26, New York, NY, USA, 2004. ACM.

[119] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

[120] F. Aurenhammer, R. Klein, and D.T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

[121] S. Hert and V. Lumelsky. Polygon area decomposition for multiple-robot workspace division. *International Journal of Computational Geometry & Applications*, 08(04): 437–466, 1998.

[122] C. H. Papadimitriou and M. Sideri. The bisection width of grid graphs. *Mathematical systems theory*, 29(2):97–110, 1996.

[123] D. Delling, D. Fleischman, A. V. Goldberg, I. Razenshteyn, and R. F. Werneck. An exact combinatorial algorithm for minimum graph bisection. *Mathematical Programming*, 153(2):417–458, 2015.

[124] A. E. Feldmann and P. Widmayer. An $O(n^4)$ time algorithm to compute the bisection width of solid grid graphs. *Algorithmica*, 71(1):181–200, 2015.

[125] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. Path planning for mobile robot navigation using Voronoi diagram and fast marching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2376–2381, Oct 2006.

[126] A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart, and D. R. Voronoi coverage of non-convex environments with a group of networked robots. In *IEEE International Conference on Robotics and Automation*, pages 4982–4989, May 2010.

[127] L. Wu, M. Angel, G. Garcia, D. P. Valls, and A. S. Ribalta. Voronoi-based space partitioning for coordinated multi-robot exploration. *Journal of Physical Agents*, 1 (1):37–44, 2007.

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

[129] S. B. Niku. *Introduction to robotics : analysis, control, applications*. Hoboken, NJ : Wiley, 2nd edition, 2011.

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

[131] S. Patel and T. Sobh. Manipulator performance measures - a comprehensive literature survey. *Journal of Intelligent & Robotic Systems*, pages 1–24, 2014.

[132] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. 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, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer Berlin Heidelberg, 2000.

[133] J. Xu, D.K. Liu, and G. Fang. An efficient method for collision detection and distance queries in a robotic bridge maintenance system. In T.J. Tarn, S.B. Chen, and C. Zhou, editors, *Robotic Welding, Intelligence and Automation*, volume 362 of *Lecture Notes in Control and Information Sciences*, pages 71–82. Springer Berlin Heidelberg, 2007.

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

[135] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[136] A. E. Carter and C. T. Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*, 175(1):246–257, 2006.

[137] M. Srinivas and L.M. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6):17–26, Jun 1994.

[138] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

[139] B. Cakir, F. Altiparmak, and B. Dengiz. Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers and Industrial Engineering*, 60(3):376–384, 2011.

[140] R. Banos, J. Ortega, C. Gil, A. Fernandez, and F. de Toro. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40(5):1696–1707, 2013.

[141] X. Gu (Gracie). The behavior of simulated annealing in stochastic optimization, 2008. M.S thesis, Iowa State University, Ames, Iowa.

[142] K. Deb and T. Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In E. Zitzler, L. Thiele, K. Deb, C. Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 67–81. Springer Berlin Heidelberg, 2001.

[143] G. Samelius, H. Andren, P. Kjellander, and O. Liberg. Habitat selection and risk of predation: Re-colonization by lynx had limited impact on habitat selection by roe deer. *PLoS ONE*, 8(9), Sep 2013.

[144] S. Wangsiripitak and D. W. Murray. Avoiding moving outliers in visual slam by tracking moving objects. In *International Conference on Robotics and Automation*, pages 375–380, May 2009.

[145] T. Stankowich and D. T. Blumstein. Fear in animals: a meta-analysis and review of risk assessment. *Proceedings of the Royal Society of London B: Biological Sciences*, 272(1581):2627–2634, 2005.

[146] J. S. Brown, J. W. Laundré, and M. Gurung. The ecology of fear: Optimal foraging, game theory, and trophic interactions. *Journal of Mammalogy*, 80(2):385–399, 1999.

[147] S. Creel, J. Winnie, B. Maxwell, K. Hamlin, and M. Creel. Elk alter habitat selection as an antipredator response to wolves. *Ecology*, 86(12):3387–3397, 2005.

[148] H. H. Viet, V. H. Dang, M. N. U. Laskar, and T. Chung. BA*: an online complete coverage algorithm for cleaning robots. *Applied Intelligence*, 39(2):217–235, Sep 2013. doi: 10.1007/s10489-012-0406-4.

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

[150] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.

[151] L. E. Parker. Multiple mobile robot systems. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 921–941. Springer Berlin Heidelberg, 2008.