Noname manuscript No.
(will be inserted by the editor)

Online Planning for Multi-Robot Active Perception with Self-Organising Maps

Graeme Best · Jan Faigl · Robert Fitch

Received: date / Accepted: date

Abstract We propose a self-organising map (SOM) algorithm as a solution to a new multi-goal path planning problem for active perception and data collection tasks. We optimise paths for a multi-robot team that aims to maximally observe a set of nodes in the environment. The selected nodes are observed by visiting associated viewpoint regions defined by a sensor model. The key problem characteristics are that the viewpoint regions are overlapping polygonal continuous regions, each node has an observation reward, and the robots are constrained by travel budgets. The SOM algorithm jointly selects and allocates nodes to the robots and finds favourable sequences of sensing locations. The algorithm has a runtime complexity that is polynomial in the number of nodes to be observed and the magnitude of the relative weighting of rewards. We show empirically the runtime is sublinear in the number of robots. We demonstrate feasibility for the active perception task of observing a set of 3D objects. The viewpoint regions consider sensing ranges and self-occlusions, and the rewards are measured as discriminability in the ensemble of shape functions feature space. Exploration objectives for online tasks where the environment is only partially known in advance are modelled by introducing goal regions in unexplored space. Online replanning is performed efficiently by adapting previous solutions as new information becomes available. Simulations were performed using a 3D point-cloud dataset from a real robot in a large outdoor environment. Our results show the proposed methods enable multi-robot planning for online active perception tasks with continuous sets of candidate viewpoints and long planning horizons.

Keywords Active perception · Multi-robot systems · Self-organising maps · Online planning

1 Introduction

Mobile robots use their sensors and perception algorithms to understand their surrounding environment. Of fundamental interest are object recognition, classification and model generation tasks, which require understanding properties such as the pose, segmentation, class and identity of a set of objects in an environment (van Hoof et al., 2014; Wu et al., 2015; Patten et al., 2016). The informativeness of observations, and therefore the performance of perception algorithms, can be improved by judiciously selecting observation locations (Chen et al., 2011). Performance can be significantly improved by using longer planning horizons (Singh et al., 2009; Becerra et al., 2016; Atanasov et al., 2014), jointly planning for multiple robots (Best et al., 2016a; Charrow, 2015; Garg and Ayanian, 2014; Xu et al., 2013;

This work was supported by the Australian Centre for Field Robotics; the NSW Government; the Australian Research Council's Discovery Project funding scheme (No. DP140104203); and the Faculty of Engineering & Information Technologies, The University of Sydney, under the Faculty Research Cluster Program. The work of Jan Faigl was supported by the Czech Science Foundation (GAČR) under research project No. 15-09600Y.

G. Best and R. Fitch
Australian Centre for Field Robotics (ACFR), The University of Sydney, NSW, 2006, Australia.
E-mail: {g.best,rfitch}@acfr.usyd.edu.au

J. Faigl
Dept. of Computer Science, Faculty of Electrical Engineering (FEE), Czech Technical University in Prague, Czech Republic.
E-mail: faiglj@fel.cvut.cz

R. Fitch
Centre for Autonomous Systems, University of Technology Sydney, NSW, 2007, Australia.
E-mail: rfitch@uts.edu.au

Hollinger et al., 2009) and considering larger sets of candidate sensing locations. However, current planning algorithms with these properties are often too computationally expensive for practical use in large scale, online and more complex active perception tasks; we propose a self-organising map algorithm as a solution to bridge this gap.

The performance of an active perception mission, such as a classification, exploration, coverage, or persistent monitoring task, is largely dependent on an appropriate choice of viewpoints. Current approaches for active perception typically estimate the value of visiting candidate viewpoints by simulating predicted observations (van Hoof et al., 2014; Wu et al., 2015; Patten et al., 2016). For complex sensor models, these predictions can be computationally expensive, which therefore restricts the capabilities of planning algorithms. Instead, we focus on planning paths for perception tasks where informative parts of the objects in the environment have been extracted. Therefore, we use an inverse sensor model to define a discrete set of overlapping continuous viewpoint regions, with associated rewards, where each part can be observed. Correlations between viewpoints can be naturally modelled in our formulation as the overlap between viewpoint regions. Figures 1 and 2 illustrate an example outdoor environment with a collection of objects observed by a 3D laser scanner. The path planning problem is to optimise the rewards gained by visiting these desirable viewpoint regions. This new formulation for active perception enables the planner to consider a continuous space of candidate viewpoints, long-horizon planning, multi-robot scenarios and efficient online replanning.

This active perception formulation describes a multi-goal path planning problem with similarities to the orienteering problem (OP) (Gunawan et al., 2016; Vansteenwegen et al., 2011) and the travelling salesman problem (TSP) (Toth and Vigo, 2001). The prize-collecting TSP with neighbourhoods (PC-TSPN) is a closely related TSP variant that has recently been solved and applied to data collection in sensor network applications (Faigl and Hollinger, 2014). In the PC-TSPN, the objective is to plan the path of a robot that maximally selects and visits a set of disks, where the objective function is defined as the sum of the path length and the rewards for visiting each disk. This objective function has convenient algorithmic properties; however, it is unclear how to balance the trade-off between path lengths and rewards when applied to real problems. Instead, we develop a new formulation that generalises the OP; we directly optimise the observation rewards while the path length is limited by a maximum travel budget. These budget constraints can be selected to meet the requirements of the application, such as fuel, time and other resource constraints, or a planning horizon.

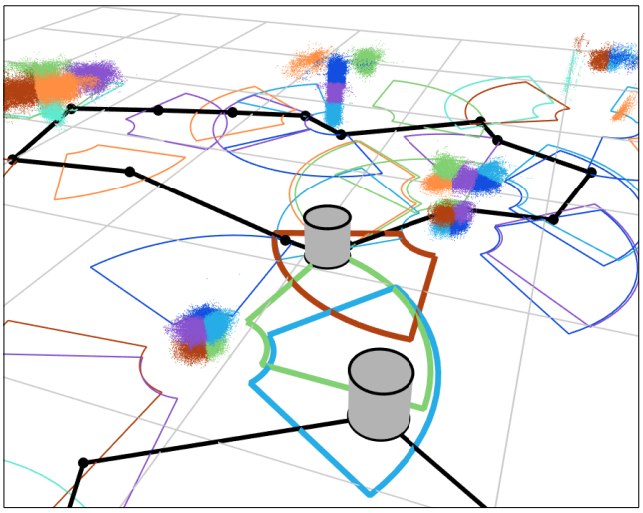


Fig. 1 Illustration of the motivating active perception problem. Each object segment (point clouds) is observed by visiting the viewpoint regions (circle segments). Grey cylinders represent positions of two robots. The currently visited viewpoint regions are drawn in bold. Black lines represent the path plans. The aim is to collectively maximise the weighted sum of viewpoint regions visited by the robots. This scene is part of the environment in Fig. 2.

The considered problem is NP-hard, which can be shown by a reduction from the orienteering problem, and therefore we turn to heuristic solutions. In particular, we consider an extension of the self-organising map (SOM) approaches for the TSP. SOM is a two-layered neural network accompanied by an unsupervised learning procedure that preserves topological properties of an input space. SOM has been applied to the traditional TSP by several authors, e.g., Angéniol et al. (1988); Somhom et al. (1997). Although SOM for the TSP does not compete with the best known combinatorial heuristics for the conventional TSP (Helsgaun, 2000), it provides a significant advantage in problems that require selecting observation locations. This is particularly important in the TSPN (Faigl and Hollinger, 2014) and the OP with neighbourhoods (Faigl et al., 2016) where the algorithm implicitly selects sensing locations within continuous neighbourhoods.

Jointly optimising the selection and sequence of nodes to observe, along with finding favourable viewpoints within sensing regions, can greatly reduce the path distance by avoiding unnecessary travel. Therefore, we consider the original idea of the SOM-based data collection planning introduced in Faigl and Hollinger (2014) for our constrained problem with limited travel budgets. Our new approach ensures these hard budget constraints are satisfied by the planning algorithm.

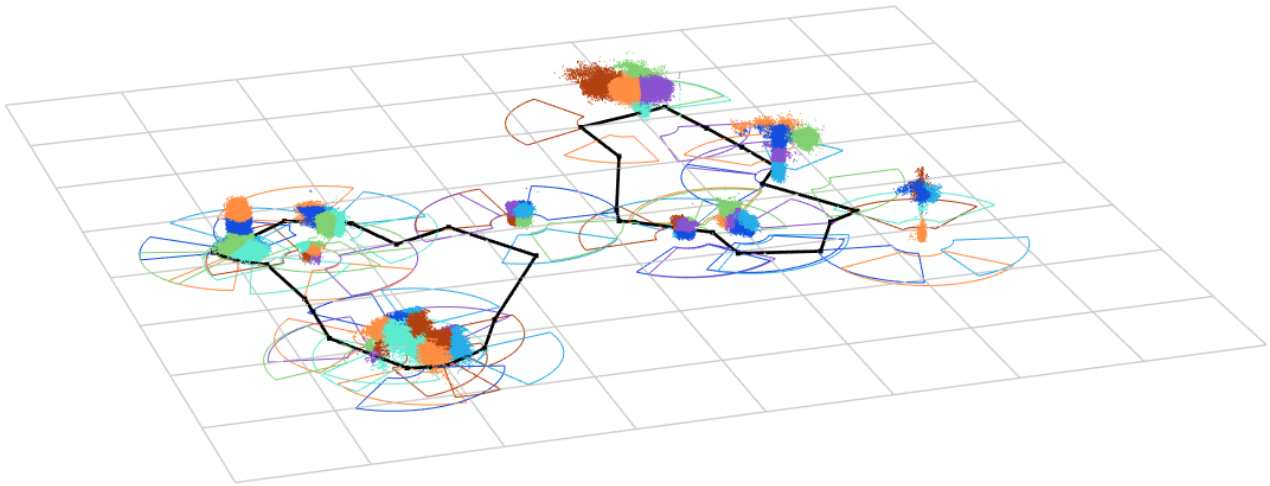


Fig. 2 An example environment, object parts, viewpoint regions and solution paths for two robots (same as Fig. 1). The 3D point cloud was generated by a real robot moving around an environment consisting of trees, tables, chairs, bins and a motorbike. The underlying grid has 5 m spacing. Almost all object parts are observed along the planned paths, with some skipped due to the travel budget constraints. In this scenario, all objects are known to the offline planner. In online scenarios, additional goals are placed in unexplored regions, and the goals and plans adapt as observations are made.

Moreover, we also generalise the approach in Faigl and Hollinger (2014) to planning for multi-robot teams. This requires addressing additional challenges, including coordinating the robots to select mutually beneficial observation locations, effectively using the available resources of each robot, and overcoming the compounded computational complexity to quickly find good solutions. Our algorithm jointly plans for multiple robots simultaneously by optimising the allocation of nodes to robots. Therefore, our approach does not require predefined or explicit partitioning of the environment. The algorithm has polynomial bounds on runtime complexity, and scales well as the number of robots increases.

A primary contribution of this paper is to demonstrate that the algorithmic approach is suitable for online scenarios. In particular, we show that the formulation can naturally incorporate exploration objectives to discover new information, and the planner efficiently performs replanning as new information becomes available. Exploration objectives can naturally be encoded as viewpoint regions, so that the planner balances between making high-quality observations of known objects and visiting unexplored space to discover new objects. This formulation is motivated by scenarios where there are two complementary sensing modalities. For example, a long-range laser sensor (Bargoti et al., 2015) detects the presence and locations of trees on a farm, while a close-range high-resolution RGB-D sensor (Martens et al., 2017; Peng et al., 2016) performs the primary task of characterising the fruit in the trees. The planner needs to balance the use of these two modalities in order for the robots to discover as many objects as possible while also making sufficient close-range observations. The proposed SOM algorithm enables efficient online replanning as new information becomes available since it is able to effectively reuse and adapt previous solutions. This is a vitally important requirement for real robots performing onboard computations while executing a mission (Likhachev et al., 2005).

In addition to theoretical analysis, we also perform simulations of several random environments and active perception tasks using a 3D point cloud dataset (Patten et al., 2015) and a realistic observation model using ensemble of shape functions descriptors (Wohlkinger and Vincze, 2011). The results highlight advantages of the algorithm in an offline setting for addressing the multi-robot, non-uniform reward, constrained budget and polygonal region characteristics of the problem. We also show the advantages of planning over continuous rather than discrete space by showing our approach outperforms the Dec-MCTS algorithm (Best et al., 2016a). Additionally we empirically evaluate the performance of the planner when incorporating exploration objectives and adapting to new information when replanning. We highlight the advantages of long-horizon planning over greedy approaches, even when limited information is available. The active perception experiments show the feasibility in practice of online long-horizon planning for multi-robot active perception tasks.

A preliminary version of this paper appeared as Best et al. (2016b). This extended version additionally contains: expanded algorithmic details; a generalised formulation for fixed start and/or end locations (similar to Faigl et al. (2016)); extended theoretical analysis of the

runtime complexity and convergence; empirical analysis of the algorithm's convergence and anytime properties; empirical validation of an example observation model definition; a comparison to Dec-MCTS; a formulation for online scenarios; and extensive simulation experiments that demonstrate the feasibility of the approach for online replanning scenarios.

The remainder of this paper is organised as follows. Sec. 2 summarises related work. Sec. 3 introduces our new problem formulation for active perception. In Sec. 4, we propose a self-organising map solution algorithm. The algorithm is analysed theoretically and empirically in Sec. 5 (and the Appendix). Secs. 6 and 7 describe how this formulation can be applied to object recognition type problems in offline and online scenarios, and show results for simulated experiments with a 3D point-cloud dataset. Finally, Sec. 8 concludes the paper and discusses avenues of future work.

2 Related Work

Active perception systems typically consist of several modules (Patten et al., 2016): a *planning* module uses the current belief of the world to select the next observation locations; a *navigation* module is responsible for driving the robot to the next chosen locations; and *observation*, *processing* and *update* modules update the belief of the world. This new belief then feeds back in to the planning module to replan the observation locations, and this process continues. Traditionally, active perception problems arise while using vision sensing modalities (Chen et al., 2011), although recently there has also been a need for new formulations suitable for sensing modalities with depth information, such as 3D laser (Patten et al., 2017), RGB-D (van Hoof et al., 2014; Patten et al., 2016; Martens et al., 2017) and thermal (Cunningham-Nelson et al., 2015) modalities.

The performance of an active perception system is highly-dependent on the observation locations selected by the planning module; in this work we develop a planner that efficiently produces high-quality sequences of observation locations and can adapt to new information collected online. In most work, the planning component of active perception systems is myopic (single-step) (van Hoof et al., 2014; Wu et al., 2015; Cunningham-Nelson et al., 2015), which is reasonable in small environments where it is assumed that previous actions do not affect the cost of future actions. Scaling the problem up to larger environments results in location-dependent action costs, and therefore performance is significantly improved by planning sequences of viewpoints over longer planning horizons. Approaches have been proposed for planning sequences of locations (Becerra et al., 2016; Atanasov et al., 2014; Hollinger et al., 2011; McMahon and Plaku, 2017), but the formulations have been limited to restricted cases, such as a single object, single robot, simple perception model, or highly-constrained action space. Recently, Best et al. (2016a) proposed the decentralised Monte Carlo tree search (Dec-MCTS) algorithm for long-horizon, decentralised, multi-robot planning. While Dec-MCTS has interesting theoretical properties and is applicable to general formulations, in this paper, we focus on developing an efficient heuristic algorithm for a more specific active perception formulation to quickly produce high-quality solution paths. Additionally, while Dec-MCTS is a parallel algorithm for decentralised planning, in this paper we are interested in scenarios where multi-robot planning is performed by a centralised processor. Another key property of our formulation is that we reason over continuous candidate viewpoint regions, which has not previously been addressed by planning algorithms for object recognition tasks. We perform an empirical comparison between our approach and Dec-MCTS in Sec. 6.3.

For active object classification and related scenarios, most planning approaches assume that the locations of objects are already known or the objects are discovered opportunistically (Atanasov et al., 2014; Hollinger et al., 2011; Wu et al., 2015; Patten et al., 2015, 2016). In large-scale environments and online planning scenarios it is necessary to incorporate exploration objectives to directly encourage discovering new objects. Once objects have been discovered, then additional observations can be made to learn higher-level semantic properties of the objects. Many approaches have been proposed for robotic exploration tasks, such as information-theoretic (Bourgault et al., 2002) and TSP (Zlot et al., 2002) formulations. Long-horizon planners for exploration typically significantly outperform greedy algorithms (Kulich et al., 2011; Faigl et al., 2012; Best and Fitch, 2016). In object classification tasks, planners should balance the trade-off between exploration objectives and the primary perception task objectives, which may by achieved using a weighted sum of the objectives (Kriegel et al., 2013; Patten et al., 2017) or multi-criteria decision making (Quattrini Li et al., 2016). The formulation we propose in this paper can naturally encode and balance exploration and primary objectives as sets of continuous viewpoint regions, which are then jointly considered by the planner.

Our new active perception problem formulation and planning approach is motivated by the work of Faigl and Hollinger (2014) for the PC-TSPN problem using an SOM algorithm. They applied the PC-TSPN to a data collection problem that requires communicating with a subset of an underwater sensor network. We

generalise this problem formulation and algorithmic approach to be more suitable for our active perception formulation. In particular, we address scenarios with multiple robots, polygonal goal regions (Faigl, 2010), budget constraints, non-uniform observation rewards, and online replanning. An SOM-based algorithm has recently been proposed for the orienteering problem with neighbourhoods (Faigl et al., 2016); this approach is also related, but we generalise for multi-robot teams, and differ in how we address budget constraints.

The orienteering problem (OP) is a type of vehicle routing problem where the paths are constrained by travel budgets. Many heuristic algorithms exist for the classic problem and related variants (Gunawan et al., 2016; Vansteenwegen et al., 2011). The most relevant variants to our formulation include the team-OP that extends the problem for multi-agent systems, generalised-OP that defines the objectives as a function of discrete sets, and the OP with neighbourhoods (OPN) where rewards are collected by visiting continuous regions. In our recent work, an SOM-based algorithm was shown to achieve comparable performance to state-of-the-art solvers for the standard OP and we extended the algorithm for the single-agent OPN (Faigl et al., 2016). For the team-OP, several approaches have been proposed, such as computationally demanding exact solvers (Dang et al., 2013a) and powerful metaheuristics based on variable neighbourhood search (Archetti et al., 2007) or particle swarm optimisation (Dang et al., 2013b). However, to the best of our knowledge, none of the existing algorithms for team-OP are capable of solving variants with continuous neighbourhoods. While OP formulations have been applied to many problems (Gunawan et al., 2016; Vansteenwegen et al., 2011), including robotics applications (Yu et al., 2016; Best and Fitch, 2016), the focus has mostly been on offline planning rather than online settings where goals are discovered over time.

The generalised-TSP (GTSP) (Noon and Bean, 1989; Smith and Imeson, 2017) is a closely related TSP variant that requires visiting at least one node in every discrete set of nodes. The GTSP has been applied to robotic path planning problems for mapping (Charrow, 2015) and mobile refuelling (Mathew et al., 2013) tasks, which are solved using a transformation to the standard TSP. Related graph-based robot path planning algorithms include Monte Carlo tree search (Best et al., 2016a), branch and bound (Singh et al., 2009; Binney and Sukhatme, 2012; Best and Fitch, 2016), recursive greedy (Chekuri and Pal, 2005; Singh et al., 2009) and sweep planes (Best et al., 2017). These formulations restrict the search to discrete points and the computation time increases with the number of points. In contrast, a set of continuous regions are efficiently searched by our proposed algorithm, and the runtime does not increase with the area of each region.

The m-TSP generalises the TSP to multiple agents, which requires assigning nodes to agents and finding a path for each agent. There are several variations of the m-TSP with different objective functions and many different approaches (Bektas, 2006; Lagoudakis et al., 2005). SOM-based approaches have been proposed for the minmax m-TSP, where the objective is to minimise the path of the longest agent. The approach creates an individual network for each agent, and the adaptation favours neurons from the currently shortest tours when allocating tasks to individual agents (Somhom et al., 1999). A similar idea has been considered for multi-agent coverage of a polygonal world with obstacles (Faigl, 2010, 2016a). However, these problems do not consider budget constraints or selecting subsets of nodes.

Our formulation is designed for active perception tasks where the rewards are viewpoint sensitive. This is particularly the case where it is required to observe a set of objects or landmarks. Our primary motivation is object classification tasks, however a variety of other tasks can be formulated in a similar way. Coverage tasks are most similar to our problem, which require a team of robots to collectively observe every location in an environment (Galceran and Carreras, 2013; Dornhege et al., 2016; Hönig and Ayanian, 2016; Bircher et al., 2016). Target tracking and search problems require using the sensing capabilities of multiple robots to locate and maintain contact with targets (Robin and Lacroix, 2015; Xu et al., 2013; Charrow, 2015). Persistent monitoring tasks require sensing an environment to reduce the uncertainty of a belief of a process (Cao et al., 2013; Garg and Ayanian, 2014). Active SLAM requires determining the relative poses of a set of landmarks (Atanasov et al., 2015). Although our problem is similar to these, in that we require dividing the workload and finding paths for multiple robots, we have a different objective function; therefore, we require a new algorithmic approach.

3 Problem Formulation

This section formally defines the budgeted multi-robot active perception problem. The objective is to plan the paths for a team of robots such that they maximally observe a set of nodes in the environment with varying rewards. Each robot has an associated travel speed and maximum travel budget. Each node may be observed by visiting any point in its associated viewpoint region, represented as a polygon. These nodes, viewpoint regions and rewards may be defined to meet the objectives of the application; in Secs. 6 and 7 we formulate example problem instances for perceiving 3D point-cloud objects

and incorporating exploration objectives. In this section we define the problem by considering the objectives that are currently known at a given time instance. Though we are interested in solving this problem in an online setting such that the plans adapt as new information is discovered, and a formulation for these scenarios is developed further in Sec. 7.

3.1 Multi-Robot Team

A team of R robots is denoted $\mathcal{R} = \{r^1, r^2, ..., r^R\}$. The trajectory of each robot r^i is defined as a sequence of waypoints $X^i = (x_1^i, x_2^i, x_3^i, ...)$, where each waypoint is a position within a free space environment $x_j^i \in \mathbb{R}^2$. Each robot r^i moves along a straight line between waypoints at a constant speed s^i, which may be different for each robot. The cost of each robot's path $c^i \geq 0$ is the time taken to travel through the sequence of waypoints X^i. Each robot has a cost budget $b^i > 0$, and a set of robot paths $\{X^i\}$ is deemed to be feasible if every robot meets its budget constraint, i.e., $c^i \leq b^i, \forall r^i \in \mathcal{R}$. We address several possible conditions for the start and end positions of the robots: i) the start and end positions are unconstrained and free to be selected by the planner, ii) the start positions are unconstrained but the robots must end at their start position, and iii) the start and/or end position is fixed.

3.2 Viewpoint Regions and Rewards

The robots aim to observe a set of N nodes $\mathcal{N} = \{n^1, n^2, ..., n^N\}$ at different locations within the environment. Every node has a weight $w^k > 0$ that defines the reward for observing the node. Each node n^k has a continuous set of viewpoints $\mathcal{Z}^k$ defined as all points on and within a simple polygon. The robot observes a node if any waypoint of the robot's path is within the viewpoint region, i.e., $\exists x_j^i \in X^i : x_j^i \in \mathcal{Z}^k$. The binary indicator variable $o^k \in \{0, 1\}$ for each node n^k is 1 if the node is observed by one or more robots and 0 otherwise. All robots sense continuously along their paths, which can be taken into account in the above definition by adding additional waypoints along a path at no extra cost. Although we assume the regions are the same for each robot, the algorithm can easily be extended to robot-dependent observations.

We are particularly interested in formulations for online tasks, where the robots should aim to observe known objectives as well as discover currently unknown objectives. This balance can be achieved using our formulation by introducing new nodes to the set $\mathcal{N}$ that represent regions where the robots may be expected to discover new objectives. We formulate this concept further in Sec. 7.

3.3 Problem Statement

The optimisation problem is to plan the locations of waypoints for each robot and the sequence the waypoints are visited X^i, such that all budget constraints are met and the sum of the observation rewards for the nodes is maximised. More formally, we aim to find the set of paths $\{X^i\}$ that optimises:

$$\begin{aligned} &\text{maximise} \sum_{n^k \in \mathcal{N}} o^k w^k, \\ &\text{s.t. } c^i \leq b^i, \ \forall r^i \in \mathcal{R}. \end{aligned}$$

We are interested in solving this problem by replanning in an online setting. The robots initially plan based on the information available offline. After each action is executed, an observation may result in a change of the nodes, viewpoint regions or rewards. We assume these changes are small and therefore the planner should efficiently adapt its previous solution to address the new objectives.

3.4 NP-hardness

The problem is NP-hard and a reduction from the orienteering problem (Vansteenwegen et al., 2011) with Euclidean costs can readily be shown by setting the number of robots R to 1 and the viewpoint sets $\{Z^k\}$ as singleton. This result motivates the development of a heuristic algorithm to approximately solve the problem in polynomial time.

4 Self-Organising Map Algorithm

Self-organising map algorithms aim to give a lower-dimensional representation of an input space, while preserving a given topological graph-based structure of the representation. For our problem the input space is the set of viewpoint regions in the environment, and the algorithm aims to find a set of sequences of waypoints (representing robot paths) that best fits this input space. The learning procedure is competitive in that each viewpoint region is presented one at a time, and each waypoint competes to be the *winner* for representing that region. A winner waypoint moves towards that region, and neighbours of the winner in the graph topology will also move towards the region by a decreasing distance. This process is repeated for a fixed number of learning

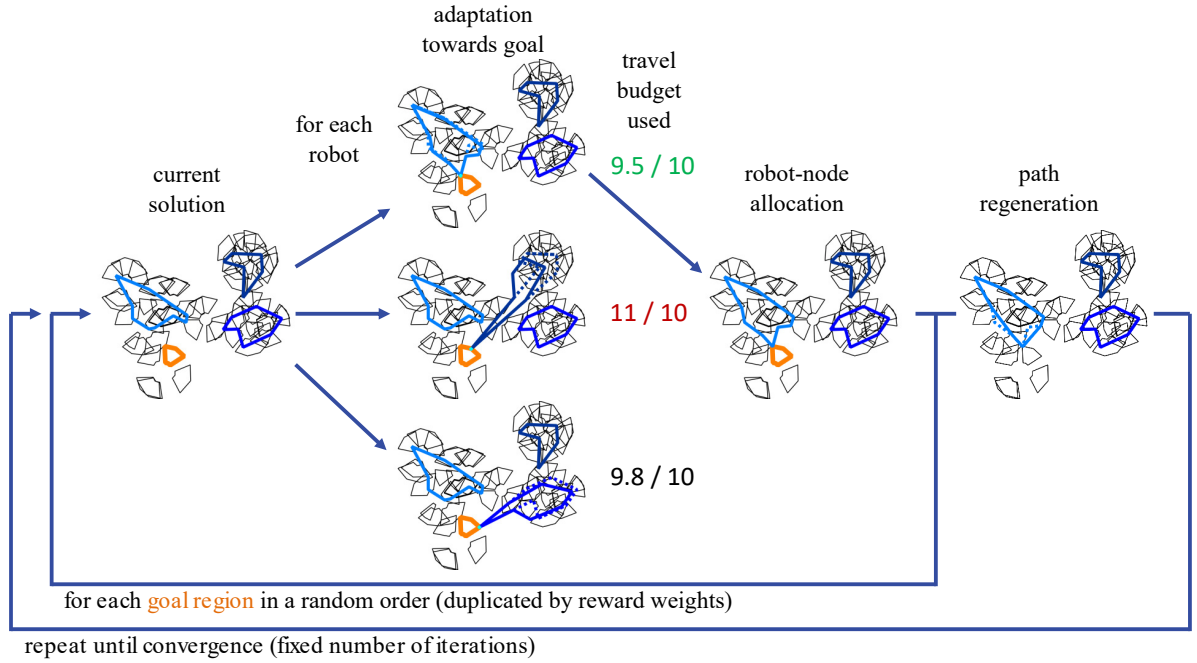


Fig. 3 Overview of the proposed self-organising map algorithm with an example problem instance for three robots.

epochs, when convergence of the paths to a stable state is guaranteed.

This section details the proposed SOM learning procedure for our problem formulation, which includes addressing non-uniform observation rewards, node selection satisfying budget constraints, multi-robot task allocation to nodes and can efficiently perform online replanning by adapting previous solutions. We first provide an overview of the algorithm followed by a detailed explanation of all components.

4.1 Algorithm Overview

An overview of the proposed self-organising map algorithm is presented in Fig. 3 and Alg. 1. The algorithm consists of two nested loops and in each iteration, the solution paths for the team of robots is adapted towards the final solution. During each iteration of the outer loop (Alg. 1 line 5), called an *epoch*, all viewpoint regions are addressed one at a time (line 7) by adapting the path of one of the robots towards the considered viewpoint region (line 12). The path adaptations are performed using an extension of the standard self-organising map adaptation process for TSP problems, in combination with a greedy robot-node allocation policy (line 11). This allocation policy divides the workload between robots while satisfying budget constraints. Regions with low reward are considered once per epoch while high-reward regions are considered multiple times. At the end of each epoch, the paths are regenerated to remove non-informative waypoints (line 13), and an adaptation parameter is cooled (line 16). The algorithm continues for a fixed number of epochs until convergence is guaranteed. During early epochs the paths typically make large sporadic jumps around the environment, while small local refinements are made in later epochs. In our analysis (later in Sec. 5) we show the runtime complexity of the algorithm is $\mathcal{O}(N'^2)$, where N' is the number of viewpoint regions after duplication to take into account the rewards $\{w^k\}$. In the remainder of this section, we provide a detailed explanation of all components of the algorithm.

4.2 Graph Topology

The graph topology for the SOM is a set of R sequences of waypoints that directly represent the robot paths. Each of these paths will transform over time according to the following learning procedure. For problems where each robot must return to its start position, the topology of each path is a closed loop. If this is not required, then the topology is a set of open paths. When performing online replanning, the paths may be initialised as the previously computed solution. If no previous solution is available, then we initialise each path as a small circle (consisting of $\lfloor N/R \rfloor$ waypoints) around the centre of a unique arbitrary node (Alg. 1 line 1). This initialisation is reasonable since the paths will quickly spread over the input space and adjust their number of nodes during the first learning epochs.

Algorithm 1 Self-organising map algorithm.

Input: robot speeds $\{s^i\}$ and budgets $\{b^i\}$,
a set of nodes $\{n^k\}$ with associated
viewpoint regions $\{\mathcal{Z}^k\}$ and rewards $\{w^k\}$,
adaptation parameters σ_0 and δ
Output: planned path for each robot $\{X^i\}^*$

1: $X^i \leftarrow$ circle around arbitrary node n^i, $\forall r^i \in \mathcal{R}$
2: $\mathcal{N}' \leftarrow$ duplicate $n^k \in \mathcal{N}$ by factor $w^k/\text{GCD}(\{w^k\})$
3: $\mathcal{N}' \leftarrow \mathcal{N}' \cup \{$virtual node for each fixed waypoint$\}$
4: $\sigma \leftarrow \sigma_0$; $i \leftarrow 1$ ▷ Adaptation parameter
5: **while** not converged **do**
6: $perm \leftarrow$ random permutation of $\{n^k\}$
7: **for each** $n^k \in \mathcal{N}'$, in order $perm$ **do**
8: **for each** $r^i \in \mathcal{R}$ **do**
9: $X^{i\prime} \leftarrow \text{ADAPTATION}(X^i, \mathcal{Z}^k, \sigma)$
10: $c^{i\prime} \leftarrow$ travel time of path $X^{i\prime}$ at speed s^i
11: $r^i \leftarrow \underset{r^i \in \mathcal{R}, c^{i\prime} \le b^i}{\text{argmin}} \left(\frac{c^{i\prime}}{b^i}\right)$ ▷ Robot selection
12: $X^i \leftarrow X^{i\prime}$ ▷ Update selected robot
13: $\{X^i\} \leftarrow$ regeneration of $\{X^i\}$
14: $F \leftarrow \sum_{n^k \in \mathcal{N}} o^k w^k$ ▷ Evaluate objective
15: **if** $F > F^*$ **then** $\{X^i\}^* \leftarrow \{X^i\}$ ▷ Save best plan
16: $\sigma \leftarrow (1 - i\delta)\sigma$; $i \leftarrow i + 1$

Algorithm 2 Adaptation step of the SOM algorithm.

function ADAPTATION
Input: path X^i of robot r^i,
viewpoint region $\mathcal{Z}^k$ of node n^k,
adaptation parameter σ
Output: an adapted path $X^{i\prime}$ for robot r^i

1: $x_\text{w} \leftarrow$ closest waypoint in X^i to $\mathcal{Z}^k$
2: $z_\text{w} \leftarrow$ closest point in $\mathcal{Z}^k$ to x_w
3: $d_\text{w} \leftarrow \|z_\text{w} - x_\text{w}\|$
4: $x_\text{e} \leftarrow$ closest point on edges of X^i to $\mathcal{Z}^k$
5: $z_\text{e} \leftarrow$ closest point in $\mathcal{Z}^k$ to x_e
6: $d_\text{e} \leftarrow \|z_\text{e} - x_\text{e}\|$
7: **if** $x_\text{w} \in \mathcal{Z}^k \vee d_\text{w} \le d_\text{e}$ **then** ▷ Winner selection
8: **if** x_w is fixed **then**
9: $x_\text{w} \leftarrow$ copy of x_w
10: $X^i \leftarrow$ insert x_w into X^i next to fixed copy
11: $x^\star \leftarrow x_\text{w}$; $z^\star \leftarrow z_\text{w}$ ▷ Select waypoint as winner
12: **else**
13: $X^i \leftarrow$ insert x_e into X^i along edge
14: $x^\star \leftarrow x_\text{e}$; $z^\star \leftarrow z_\text{e}$ ▷ Select edge as winner
15: **for each** $x^i_j \in X^i$ **do**
16: **if** x^i_j is not fixed **then**
17: ▷ Adapt waypoints in neighbourhood of $x^\star$
18: $l \leftarrow$ cardinal distance from $x^\star$ to x^i_j
19: $x^i_j \leftarrow$ move x^i_j towards $z^\star$ by factor f in Eqn. (1)

4.3 Viewpoint Rewards

Each node has an associated reward for being visited. To ensure that the learning procedure favours visiting the higher reward viewpoint regions, each node is duplicated according to its reward. The node n^k is duplicated by a factor of w^k divided by the greatest common divisor of the set of rewards $\text{GCD}(\{w^k\})$. This is performed in Alg. 1 line 2. The computation time complexity is dependent on the number of duplications. Therefore it may be beneficial to reduce the number of duplications by rounding the rewards to the nearest multiple of a number greater than $\text{GCD}(\{w^k\})$.

The motivation for this approach is that high-reward regions will be trialled more often in each learning epoch. This increases the likelihood of a robot path transforming towards the higher weighted nodes, decreases the likelihood of the node not being selected due to budget constraints, and decreases the likelihood of waypoints in high-reward regions being removed during the regeneration step.

4.4 Learning Epochs

In each learning epoch (iteration of Alg. 1 line 5 loop), each node is considered one at a time, and one robot is selected to transform its path towards each viewpoint region, if it meets its budget constraint. At the end of each learning epoch, any unnecessary waypoints are removed before starting the next learning epoch. We describe these steps in more detail as follows.

Permute the nodes

At the start of each epoch, the nodes are permuted in a random order which will determine the order that they are considered (Alg. 1 lines 6-7). This ensures the algorithm is less sensitive to the ordering and the initial conditions, and more likely to escape from local optima.

Winner selection and adaptation

The key steps in the SOM algorithm are the winner waypoint selection and the adaptation of the position sequences, as detailed in Alg. 2. For every node, this is performed for each robot, but then in the following step a robot allocation policy ensures only one of the robots gets updated for each node. The winner waypoint selection is performed by considering all waypoints and edges in the current robot path X^i. The existing waypoint or a point along one of the existing edges that is closest to any point within the viewpoint region $\mathcal{Z}^k$ is considered as the winner (Alg. 2 line 11 or 14). If the winner is a point along an edge, then a new waypoint is inserted into the path at this point (line 13).

The winner waypoint $x^\star$ is then moved to the closest point $z^\star$ in $\mathcal{Z}^k$. If $z^\star$ is on the edge of the polygon it is moved slightly towards the centre to avoid numerical issues and to reduce the chances of the waypoint being immediately removed in the next path regeneration phase. The cardinal distance (number of hops) in the path/loop from $x^\star$ to every other existing waypoint is

denoted l. Each waypoint in X^i is moved some fraction towards $z^\star$ (line 19), such that waypoints with low cardinal distance to $x^\star$ will move further towards $z^\star$ than other waypoints. This fraction is determined by the neighbourhood function

$$f(\sigma, l) = \begin{cases} \mu e^{-\frac{l^2}{\sigma^2}} & \text{for } i < 1/\delta \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where i is the current learning epoch and the gain decreasing rate δ is a small constant parameter, e.g., $\delta = 0.002$. The value of σ is decreased at the end of each learning epoch as $\sigma \leftarrow (1 - i\delta)\sigma$ (Alg. 1 line 16), which causes the neighbours to adapt less as the algorithm progresses. We define the learning rate as $\mu = 1$ throughout this paper; alternatively, a cooling schedule could also be defined for μ (Zhang et al., 2006).

Forcing $f(\sigma, l)$ to 0 after $i = 1/\delta$ is a natural restriction of the neighbourhood function since $\sigma = 0$ when $i \geq 1/\delta$, which would cause an undesirable division by zero. According to this definition (1), there is a maximum number of epochs $i_{max} = 1/\delta$ before the adaptations stop and therefore the network converges. For example, $\delta = 0.002$ provides $i_{max} = 500$. We discuss the convergence properties further in our Sec. 5.1.1 analysis and in the Appendix.

Robot-node allocation

After adapting each path towards the viewpoint region $\mathcal{Z}^k$, the algorithm then only allocates one (or none) of the robots to the node and only this robot keeps their adapted path. The selection is performed by greedily selecting the robot that has used the least fraction of its budget after performing the adaptation (Alg. 1 line 11). If no robot meets their travel budget then no paths are adapted. It is important to note that this allocation of robots to nodes is greedy just for the currently presented node and the particular learning epoch; these allocations are often modified in later learning epochs if a better allocation is found, and thus the overall SOM algorithm is not a greedy algorithm.

This allocation approach is motivated by the observation that in most cases an optimal solution should have each robot using approximately all of its travel budget. This is similar to what is typically seen in minimax problems. We wish to divide the work evenly between the robots as the learning progresses towards the final solution, such that a natural partitioning is found between the robots. Conversely, unbalanced path growth is likely to result in poor partitioning, such as when planning for the robots sequentially (as seen in the experiments in Sec. 5.2.1).

Other possible allocation policies could also be appropriate here, such as the Hungarian algorithm. However, we believe it is better to have a fast and simple allocation policy, such as the greedy policy. This is because the actual reward or cost of each allocation is difficult to measure due to the flow-on effect of optimising *sequences* of viewpoints. Heuristic approaches are therefore appropriate, and suboptimal allocations can be quickly modified again in later epochs.

Path regeneration

At the end of each epoch (Alg. 1 line 13), waypoints that are no longer useful are removed. A waypoint is useful if it is within at least one of the viewpoint regions. If multiple waypoints are within a viewpoint region then only one waypoint is randomly selected to remain, since there is no additional reward for multiple observations of a node. This also ensures the cardinal length of the paths do not grow beyond N and therefore the computation time complexity at each iteration is bounded (see Sec. 5).

If all waypoints for a robot are removed, then the robot's path is reinitialised following the procedure described in Sec. 4.2 with a new unique arbitrary node. This will not occur regularly in non-trivial problems since the robot will typically be allocated at least to its arbitrary starting node.

Start and end conditions

If the problem formulation specifies fixed start and/or end positions for the robots, then fixed waypoints are added to each path at these positions. If any of the fixed waypoints are selected as a winner, then the waypoint is duplicated (Alg. 2 lines 9-10) and the new waypoint is adapted instead of the fixed waypoint (line 11). Fixed waypoints are not moved during the neighbourhood adaptations (line 19). Additionally, during each epoch, an adaptation is performed towards each fixed node—equivalent to if there was a singleton viewpoint region at each fixed waypoint (Alg. 1 line 3). This ensures the waypoints with low cardinal distance to the fixed nodes maintain a minimal Euclidean distance to neighbouring fixed nodes. Fixed waypoints cannot be removed during the path regeneration step.

Adaptation parameter

The attraction between neighbouring waypoints during each adaptation is dependent on the σ parameter of the neighbourhood function (Alg. 2 line 19). When σ is large then several waypoints will typically move a large distance during each adaption and therefore large global

adjustments are made to the solution paths. Conversely, when σ is small then only waypoints within a small neighbourhood of the winner will move and therefore only small local refinements are made to the solution paths. Convergence of the algorithm is controlled by initialising σ to an input parameter σ_0 and then cooling σ after each epoch at a rate determined by an input parameter δ (Alg. 1 line 16). Therefore, the number of epochs before the solutions reach a steady state are determined by σ_0 and δ.

In online scenarios, new observations will typically cause minor adjustments to the objective function and therefore only minor local refinements to the previous solution are required. Online replanning can therefore be performed more efficiently by using the previous solution paths as the initial paths and initialising σ to a lower σ_0 value. We demonstrate suitability for online replanning in the Sec. 7 experiments.

5 Analysis

This section provides a theoretical analysis of the algorithm's runtime complexity and convergence, and then empirical analysis of the behaviour of the algorithm for various random environments. Further experiments are shown later in Secs. 6 and 7 that focus on active perception of 3D point clouds in offline and online scenarios.

5.1 Theoretical Analysis

5.1.1 Runtime complexity

The runtime complexity of the algorithm is polynomial in the number of nodes to be observed and the magnitude of the relative weighting of rewards. We formally state and prove this result as follows. Lemma 1 states the runtime complexity for each epoch. Lemma 2 states the maximum number of epochs is constant, assuming a given cooling schedule. These results are combined in Theorem 1 to state the runtime complexity of the algorithm. We then remark on implications of this result. Further analysis of the convergence properties is provided in the Appendix.

Lemma 1 *The runtime complexity for each epoch is upper bounded by*

$$\mathcal{O}\left(\left(\sum_{i=1}^{R}|X^i|\right)N'\right) \leq \mathcal{O}\left(N'^2\right) = \mathcal{O}\left(N^2\left(\frac{\text{MAX}(\{w^k\})}{\text{GCD}(\{w^k\})}\right)^2\right),$$

where $|X^i|$ is the number of waypoints in the path for robot i, N is the number of viewpoint regions and N' is the number of viewpoint regions after duplication to take into account the rewards $\{w^k\}$.

Proof The ADAPTATION function (Alg. 2) has runtime $\mathcal{O}(|X^i|)$, where $|X^i|$ is the number of waypoints in the current path for robot i. In the inner-loop of Alg. 1 (lines 8-10), ADAPTATION is called once per robot, and thus the runtime for lines 8-10 is $\mathcal{O}(\sum_{i=1}^{R}|X^i|)$. Since only one waypoint is allocated to a node during each epoch, and the regeneration step removes all waypoints not allocated to a node at the end of each epoch (Alg. 1 line 13), it holds that $\sum_{i=1}^{R}|X^i| \leq N$ at the end of each epoch. At most N' new waypoints are added during each epoch (if all winners are edges), and thus $\sum_{i=1}^{R}|X^i|$ is upper bounded by $N + N'$. Thus, the runtime for the line 8 loop is bounded by $\mathcal{O}(N + N') = \mathcal{O}(N')$.

In each epoch, this is repeated for each duplicated node (N') and any fixed start or end nodes (up to $2R$, if applicable). Thus, the runtime for each epoch is bounded by $\mathcal{O}((\sum_{i=1}^{R}|X^i|)(N' + R)) \leq \mathcal{O}(N'(N' + R))$. The R term only exists for problem instances that specify fixed waypoints for start and end conditions. Furthermore, $R \ll N \leq N'$ for non-trivial problems; therefore, the R term is negligible. Thus, the runtime for each epoch is bounded by $\mathcal{O}(N'^2)$.

Each viewpoint region is duplicated up to $\frac{\text{MAX}(\{w^k\})}{\text{GCD}(\{w^k\})}$ times and thus $N' \leq N\frac{\text{MAX}(\{w^k\})}{\text{GCD}(\{w^k\})}$. Therefore, $\mathcal{O}(N'^2) = \mathcal{O}(N^2(\frac{\text{MAX}(\{w^k\})}{\text{GCD}(\{w^k\})})^2)$. □

Lemma 2 *The algorithm is guaranteed to converge within $i_{max} = 1/\delta$ epochs, where the gain decreasing rate δ is a fixed parameter of the algorithm.*

Proof The neighbourhood function $f(\sigma, l)$, as defined in (1), will become 0 for all l when the number of learning epochs $i \geq 1/\delta$. When this occurs, all of the waypoints will remain at their current positions and therefore the network will not evolve any further. □

Theorem 1 *The runtime complexity of Alg. 1 is upper bounded by*

$$\mathcal{O}\left(\left(\sum_{i=1}^{R}|X^i|\right)N'\right) \leq \mathcal{O}\left(N'^2\right) = \mathcal{O}\left(N^2\left(\frac{\text{MAX}(\{w^k\})}{\text{GCD}(\{w^k\})}\right)^2\right),$$

where $|X^i|$ is the number of waypoints in the path for robot i, N is the number of viewpoint regions and N' is the number of viewpoint regions after duplication to take into account the rewards $\{w^k\}$.

Proof Lemma 2 states the maximum number of epochs is constant, and thus the runtime complexity is a constant multiple of the epoch runtime given in Lemma 1. □

Remark 1 (Runtime dependence on R) Interestingly, the derived upper bound on runtime $\mathcal{O}(N'^2)$ does not directly depend on the number of robots R, and is instead dominated by properties of the environment. The key to the derivation of this bound is that each viewpoint region is allocated to a maximum of one robot during each epoch, and therefore the *maximum* total number of waypoints is independent of R. This results in the line 8 loop having a runtime bounded by $\mathcal{O}(N')$, as described in the proof of Lemma 1, which does not directly depend on R. If, in an alternative algorithm, more than one robot could be allocated to a node, the line 8 loop runtime bound would increase to $\mathcal{O}(RN')$, which is instead linear in R.

However, it is important to note that the tighter bound $\mathcal{O}((\sum_{i=1}^{R}|X^i|)N')$ is linear in $\sum_{i=1}^{R}|X^i|$. Thus, if the team plans to observe a larger number of nodes, then the runtime will increase. There are several contributing factors that affect the number of observed nodes, including the number of robots R, the travel budgets, the fixed start and end positions, and the distribution of nodes and rewards in the environment. Importantly, the number of observed nodes, and therefore the runtime, will typically be sublinear in R, which we confirm empirically in Sec. 5.2.5. △

Remark 2 (Early convergence) Lemma 2 defines an upper bound on the number of epochs; though, in practice, convergence will typically occur much sooner than i_{max} epochs. Early convergence occurs for a number of reasons, which we summarise here, and elaborate on further in the Appendix. Empirical evidence of convergence is provided in Sec. 5.2.4. Related discussions of convergence may be found in Cochrane and Beasley (2003); Faigl and Hollinger (2017); Tucci and Raugi (2010).

Most importantly, for the neighbours of the winner (i.e., $l > 0$), the neighbourhood function $f(\sigma, l)$ pragmatically becomes zero much sooner than epoch i_{max}. For example, when using IEEE 754 arithmetic, with $\sigma_0 = 4$ and $\delta = 0.002$ (therefore $i_{max} = 500$), the neighbourhood function becomes zero for $l > 0$ at epoch $i = 68$. When this point is reached, the winners $x^\star$ are adapted with $f(\sigma, 0) = 1$, but the neighbours are never adapted. It is possible for the winners $x^\star$ to continue adapting until epoch i_{max}, however this is unlikely to occur due to the travel budgets being exhausted.

Furthermore, our SOM algorithm maintains the *best* solution $\{X^i\}^*$ at the end of each epoch (Alg. 1 line 15), which is likely to converge before the network $\{X^i\}$ converges. This is because the network may oscillate between different nodes due to the random permutation of n^k (Alg. 1 line 6), while the best found solution remains constant. △

5.1.2 Optimality

Self-organising map algorithms, including ours, are stochastic learning procedures that can guarantee convergence in polynomial time, but unfortunately cannot guarantee optimality in finite time. These algorithms therefore are heuristic algorithms for giving approximate solutions to NP-hard problems in polynomial time. The algorithm does however have the advantage of being anytime, i.e., the algorithm can be halted early, since all intermediate solutions are feasible solutions. The parameters σ_0 and δ can also be tuned to strike a balance between optimality and computation time, and we exploit this property in the Sec. 7 formulation for online scenarios. The computation time can also be reduced, potentially at the cost of solution quality, by reducing $\frac{\text{MAX}(\{w^k\})}{\text{GCD}(\{w^k\})}$ by rounding the rewards to multiples of a divisor greater than $\text{GCD}(\{w^k\})$.

5.2 Empirical Analysis

Simulated experiments were performed to analyse the behaviour of the algorithm under various conditions. Since the problem is new, we do not have algorithms for direct comparison. Therefore, we compare to restricted versions of our algorithm with some components removed to analyse how the various algorithmic components contribute to generating high-quality solutions. We compare i) planning using the joint multi-robot optimisation compared to sequential optimisation, ii) planning with and without the viewpoint rewards, and iii) planning with the viewpoint polygons compared to singular points. We also demonstrate the convergence and anytime properties. The algorithm plans paths through 100 random environments consisting of random sets of polygons. An example environment is illustrated in Fig. 4.

The parameters are as follows, except where varied for specific experiments. The environments are a continuous 1000×1000 space. There are 80 polygons with random centre points and from 3 to 6 vertices spaced at equal angles around the centre. The distance from the centre to each vertex is random between 40 and 120. Rewards are exponentially distributed between 1 and 4 and rounded to the nearest integer, such that few regions have high rewards. There are 3 robots with budgets 800, speeds 1 and a closed-loop path topology with free start locations. In all cases, convergence was reached in 70 epochs. The same sample environments

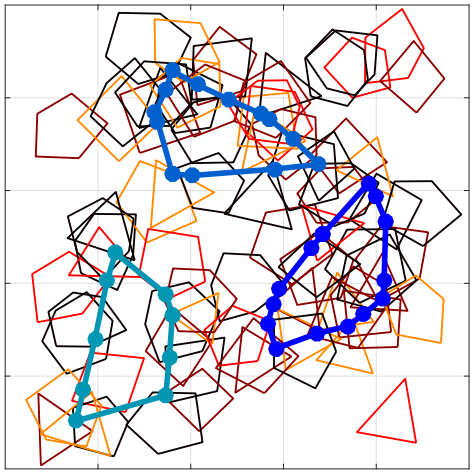


Fig. 4 Example path plans for three robots (blue) through a set of random viewpoint regions weighted from 1 (black) to 4 (orange). The robots visit a weighted sum of 155 viewpoint regions out of a maximum 170. Each robot has a budget of 1000 and speed 1.

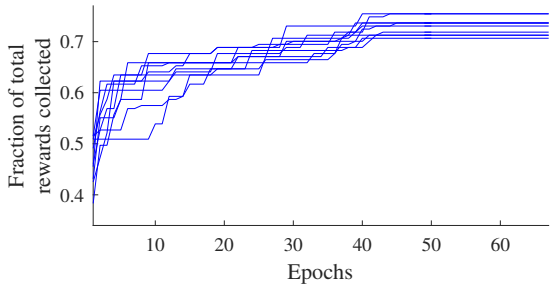


Fig. 5 Convergence of the best solution found by the algorithm for 10 trials of a single random problem instance.

are used for each pair of methods and a single-tailed paired t-test was performed for each comparison. For these experiments we use $\sigma_0 = 1$ and $\delta = 0.001$.

5.2.1 Multiple robots

Fig. 6a shows the rewards collected by planning using the proposed method, which jointly optimises multiple robots, compared to planning for the robots sequentially. The sequential method performs the SOM algorithm for a single robot at a time, with each robot ignoring the nodes selected by previous robots. The two methods were compared for 2 to 6 robots, where the budgets were uniform and summed to 2400. The simulations show the proposed approach has the best performance in all cases, and these results were statistically significant ($p < 0.01$) in all cases except $R = 4$. The largest improvements were for planning for smaller teams, because in these cases the performance is greatly influenced by effective partitioning of the workspace between the robots, which can be more effectively optimised when planning for all robots jointly.

5.2.2 Observation rewards

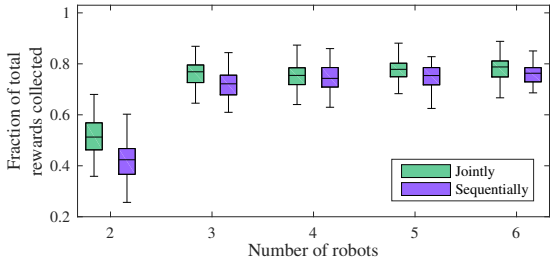
Fig. 6b shows the simulation results for planning using the proposed duplication approach compared to assuming uniform rewards. The rewards are exponentially distributed between 1 and $\bar{w}$ with lower rewards more likely, and $\bar{w}$ varied from 2 to 32. For this comparison method, the non-uniform rewards are not known to the planner, but the resulting solution paths are evaluated with respect to the non-uniform reward model. These experiments were performed with a budget of 600 and an average polygon size of 40. In all cases, planning with the proposed approach improved the performance, and these results were statistically significant ($p < 0.01$). Greater improvements were achieved when the maximum reward was large since the proposed approach is more likely to select nodes with large rewards.

5.2.3 Viewpoint regions

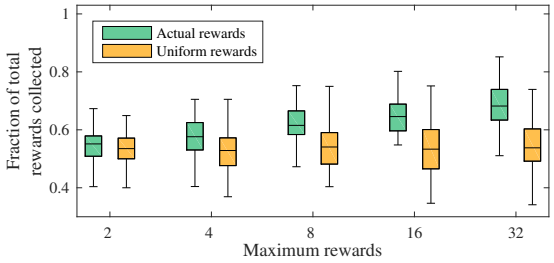
We analyse the value of the proposed planning with continuous polygonal viewpoint regions compared to planning with single points at the region centres. Fig. 6c compares these two methods with a varying number of nodes and Fig. 6d has a varying average polygon size. The proposed planner outperformed the single point planner for all number of nodes and when the polygon size ≥ 20, and these results were statistically significant ($p < 0.01$). When the polygon size was very small (10) it was sufficient to plan by approximating the polygons as single points. The proposed approach achieved greater improvements when the number of nodes and the size of the polygons were large. In these cases, the algorithm can more effectively take advantage of being able to optimise the waypoint locations.

5.2.4 Convergence

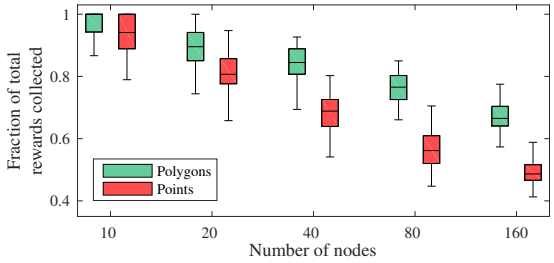
In Fig. 5 we illustrate the convergence of the algorithm for repeated trials of a single random problem instance. In all trials, the intermediate solutions made incremental improvements and converged towards the final solution, which was reached before 45 epochs. This convergence demonstrates that the algorithm is anytime since each intermediate solution is a feasible solution. This is an important property in practical applications where the computation budget is not known in advance, and therefore the algorithm may need to be halted early and return the best solution found so far. If the computation budget is known in advance, then the parameters may be



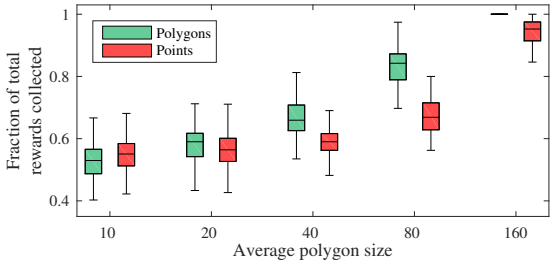
(a) Jointly planning for all robots following the proposed method compared to sequentially planning each robot.



(b) Planning while considering the actual observation rewards compared to planning assuming uniform rewards.



(c) Planning while considering the viewpoint regions compared to planning for only the centroid of the polygons.



(d) Planning while considering the viewpoint regions compared to planning for only the centroid of the polygons.

Fig. 6 Simulation results for random environments under various scenarios and comparison methods. Vertical axes shows performance as the ratio of the achieved weighted sum of nodes visited to the weighted sum of all nodes in the environment. Box plots show lower bound, lower quartile, median, upper quartile, and upper bound for 100 sample environments.

tuned to meet this requirement; we discuss this idea further in Sec. 7.2.4. We provide further insight regarding the convergence of the algorithm in the Appendix.

5.2.5 Computation time

The SOM algorithm was implemented in MATLAB and the simulations were performed on a standard desktop computer with an Intel i7 processor on a single core. The runtime varied from 0.5 s to 30 s depending on the scenario. The trends agreed with the theoretical analysis such that runtime increased with the number of nodes and maximum weight. The runtime increased sublinearly with the number of robots, which agrees with our analysis in Remark 1. Runtime was dominated ($\approx 70\%$) by the winner selection and the waypoint usefulness evaluation, since these geometric computations are relatively expensive. Our implementation has not been thoroughly optimised since our primary focus was on validating the feasibility of the approach. Therefore, the runtime can be significantly improved by the implementation, as well as by using approximations, such as decreasing the number of polygon vertices or approximating polygons as discs.

6 Active Perception of 3D Point-Cloud Objects

Our primary motivation for the proposed problem formulation and SOM algorithm is active perception tasks that aim to observe a set of object parts in a large environment. These problems rely on prior observations or a predefined belief of the environment, which may have come from a coarse scan with noisy sensors. The aim is now to perform a more informative or complete scan of the environment, and this process may be repeated. In this section, we demonstrate how the algorithm can be applied to this class of active perception tasks. For these experiments we assume the observation regions and rewards are known in advance by an offline planner, while in Sec. 7 we extend the formulation for closed-loop scenarios where this information is discovered online.

We consider example scenarios using three variations of an outdoor scene from a real 3D point-cloud dataset first presented in Patten et al. (2015). The data was recorded with a Velodyne laser scanner mounted on a robot pictured in Fig. 7. Observations were made from several locations and fused together. The three scenes consist of 12, 15 and 18 objects spread around a 40 m×40 m environment, including trees, tables, chairs, bins and a motorbike. The dataset has been used previ-

Fig. 7 The robot moving through the environment and using its onboard Velodyne laser scanner to collect the 3D point-cloud dataset (Patten et al., 2015).

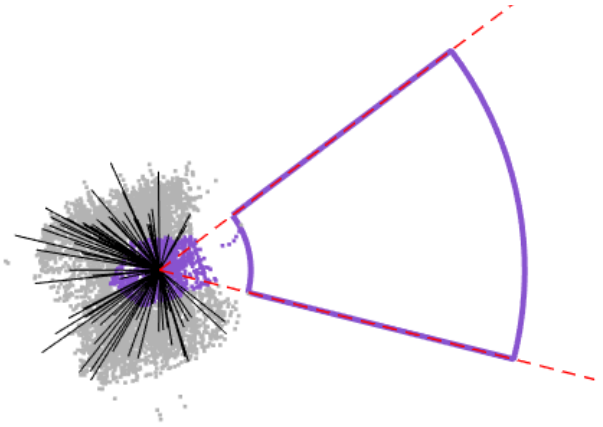


Fig. 8 Illustration of a viewpoint region (purple shape) for an associated object part (purple point-cloud), defined using the example sensor model. The point cloud represents observations of a table object, depicted from above. An object part is highlighted as a purple point cloud. The black lines are a subset of the vectors representing self-occlusions between the purple part and the rest of the object. Dashed red lines define the viewing angle.

ously for testing object classification algorithms (Patten et al., 2015; Best et al., 2016a; Patten, 2017).

The environment is represented by a set of parts in a 3D point cloud with associated viewpoints and rewards. Examples of the segmentation and viewpoint regions are shown earlier in Figs. 1 and 2. The point cloud processing is summarised as follows: i) oversegment the environment into parts, ii) estimate self-occlusion free viewpoint regions for each part, and iii) define the rewards as the discriminability between parts. We define this point cloud processing in more details in Sec. 6.1. Empirical validation of this model is presented in Sec. 6.1.1.

Our general objective function formulation provides a convenient way of expressing the viewpoint sensitivity of perception algorithms. The perception model defined here is an example instantiation of the viewpoint regions and rewards, and is intended to be generic for the purpose of evaluating the performance of our proposed planning algorithm. We emphasise that the proposed SOM algorithm is not limited to this perception model, but rather the model can be adapted to suit the requirements of a perception task.

6.1 Observation Model for 3D Point Cloud Objects

The point cloud of the environment is segmented into parts by removing the ground plane and then segmenting into objects using region growing. Each object is oversegmented into 5 parts using k-means clustering on the set of 3D points associated with the object.

A viewpoint region is defined for each part by considering the sensing range, as well as occlusions caused by other parts of the object. These viewpoint regions could be computed in many different ways, but we describe our implementation for these experiments as follows. An illustration is provided in Fig. 8 for computing the viewpoint region (purple shape on right) associated with an object part (purple point cloud on left). First, we compute the set of vectors (black lines), which represent occlusions. These vectors are from all points within the object part to all points in other parts of the same object (grey point cloud). Any of these vectors that have a *vertical* angle outside the range of $-\pi/8$ to $\pi/8$ are removed since they are unlikely to represent an occlusion. Next, the horizontal angles of all the remaining vectors are considered to represent occluded angles. Then, we find the largest window of angles that contains less than 10% of the occluded angles. The viewing angle range (between dashed red lines) is defined as the middle third of this window. The useful sensing range is defined as 1 to 4 m. The viewpoint region (purple shape) is defined as the intersection of the horizontal viewing angle range and the sensing range, measured relative to the part's centroid. For efficiency, this region is approximated by a polygon with 6 to 8 vertices.

We define the rewards as the discriminability of each part in a feature space. Parts with a higher discriminability contain more unique features and therefore are more likely to provide useful information to an object classifier. To measure discriminability, we perform feature extraction for each part, calculate the distance to all other parts in feature space, and normalise for each object. We compare each part to all other parts in the environment; alternatively each part could be compared to an object library. For the feature extraction, we use the ensemble of shape functions (ESF) global feature descriptor (Wohlkinger and Vincze, 2011), which

is commonly used for object classification tasks (Patten et al., 2016; Wohlkinger et al., 2012). Discriminability is measured as the exponential of the sum of Mahalanobis distances in the feature space between each part and every other part. Each object is considered to be equally important, and therefore the sum of rewards for each object is normalised to 10. Each reward is rounded to the nearest integer. The rewards for the datasets ranged from 1 to 10 with 1 or 2 more likely.

6.1.1 Model Validation

Here, we provide a short validation of this example model instantiation by illustrating how it maps to an existing perception technique. In particular, we show how it maps to an existing object recognition perception model (Patten et al., 2016), which is an instance of the general framework in Wohlkinger et al. (2012).

We implement a simplified version of the model by Patten et al. (2016) as follows. First, we build an offline database of object models, and then an observed object is probabilistically classified as an instance of an object in the database. The database is built by making several point cloud observations of each object from different angles. For each observation, the ESF global feature descriptor is stored. To classify an observed object, the ESF descriptor of the observed point cloud is computed and the Mahalanobis distance is measured to each database object and viewpoint. For each database object, the distance to the viewpoint with the closest distance is stored. A probability distribution is defined over the set of objects by computing a negative exponential of the closest distances and normalising. Multiple observations of the same object from different viewpoints are fused using Bayes' rule. The objective function is the *total entropy*, defined as the object classification entropy summed over all objects.

For this comparison we use the high-clutter dataset and generate a set of 100 single-robot paths with varying reward. The random paths were generated by running the SOM algorithm with randomly varying parameters. The utility of each path was evaluated using the example objective function in Sec. 6.1. For the comparison, the recognition performance is estimated using the point cloud observations in the dataset at viewpoints that are closest to each path.

The results are shown in Fig. 9. There is a clear correlation between the rewards computed using the proposed formulation and the comparison model (linear trendline has $r^2 = 0.67$). The correlation is strong enough to indicate there is a reasonable mapping between the two models, and that the viewpoint region definition is a suitable model for evaluating the performance of our planning algorithm.

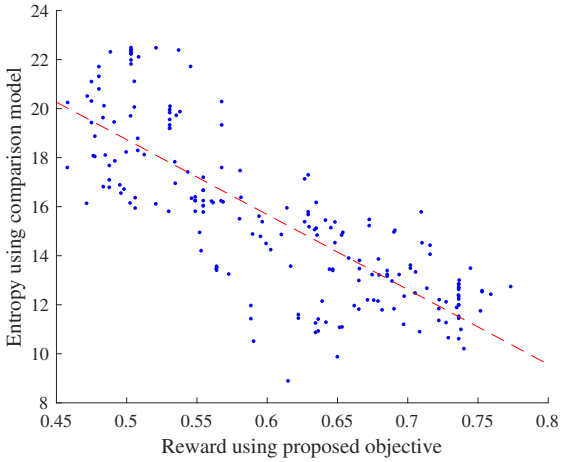
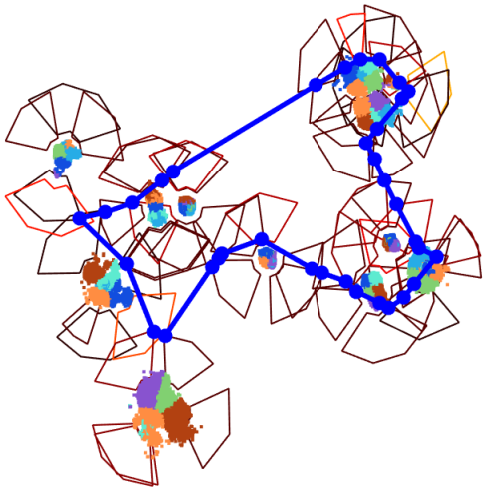


Fig. 9 Comparison of path utility between the example perception model defined in Sec. 6.1 suitable for the SOM formulation (horizontal axis) and the total entropy when using an existing object recognition model described in Sec. 6.1.1 (vertical axis). Evaluated for 100 random paths generated with SOM algorithm. Linear trendline shown in red.

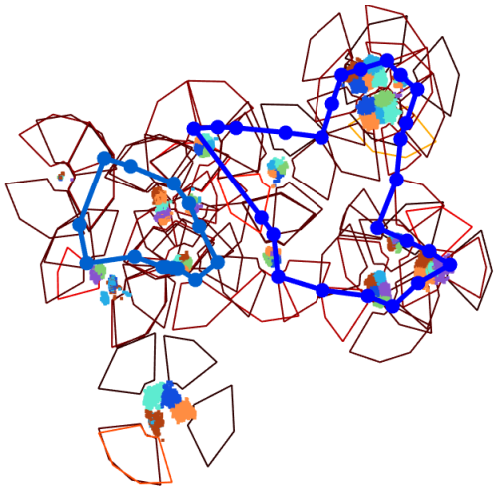
6.2 Results

We analyse four example scenarios illustrated in Fig. 10, for three environments with varying clutter. In the scenarios, we plan for: i) a single robot in the low clutter environment, ii) two robots in medium clutter, where one robot has double the budget, iii) three robots in high clutter, where the robots have speeds 2, 1.5 and 1, and iv) five robots in high clutter, where the robots have equal speeds and budgets. In these scenarios, the start positions of the robots are unconstrained but the robots must end at their start position; scenarios with fixed start positions are trialled later in Sec. 7. Planning was repeated 100 times each to measure the planning consistency.

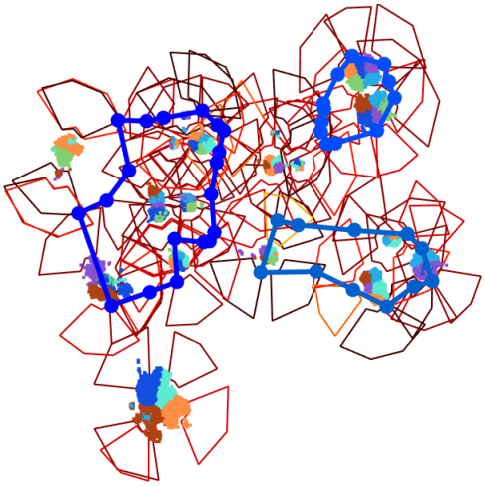
In the first scenario, the robot observed a weighted sum of 132 nodes, averaged over 100 trials, out of the maximum possible 151 nodes. The performance was consistent over the 100 trials, with a standard deviation of 2.13 weighted nodes. The worst plan had 124 and the best had 135. The average runtime was 6 s with standard deviation 0.1 s. An example solution is shown in Fig. 10a. All objects have at least one of its parts observed. The parts not selected were in the bottom left and top left, which is expected since the time to travel to these regions is relatively high. The waypoints within the selected regions naturally found locations near the edges of the regions and closer to the other



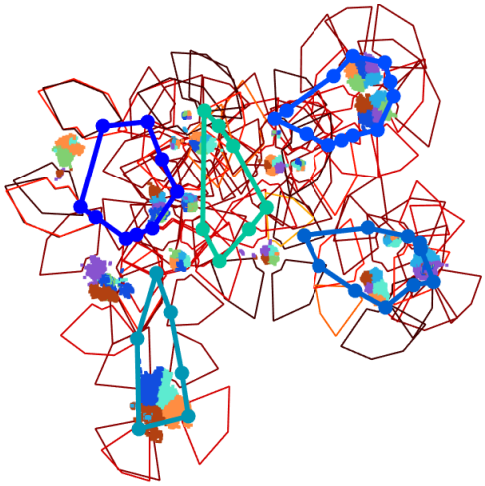
(a) Low clutter (12 object) environment with a single robot.



(b) Medium clutter (15 object) environment with two robots. Robot on right has 2× budget.



(c) High clutter (18 object) environment with three robots. Robot on left has 2× speed; bottom right has 1.5× speed.



(d) High clutter (18 object) environment with five robots. Robots have equal budget and speed. In this scenario, full coverage is achievable.

Fig. 10 Four example active perception scenarios and solution paths (blue) for varying number of robots. Object parts are shown in the coloured point clouds. Viewpoint regions are coloured black (low reward), orange (medium) and yellow (high).

regions, which implicitly minimises the travel time. All of the parts in the top right were selected, even though they are further from the other objects, since there is a significant reward to be gained by visiting two objects in close proximity.

The second scenario was planned for two robots with different budgets. Fig. 10b shows that the algorithm finds a natural partitioning between the robots in the same ratio of the travel budgets. The implicit partitioning naturally shared some of the objects between the two robots where the object parts were closer to a different robot. The planner typically avoided the object in the bottom left since there is a significant travel cost to reach those regions. The 100 trials had a weighted sum of 174 nodes on average, with a standard deviation of 3.4, out of the maximum 189. The worst plan had 156 while the best had 177, showing the distribution of plans was skewed towards the best performing plans. The average runtime was 11.7 s with the standard deviation 0.2 s.

A similar partitioning was achieved in the third scenario, shown in Fig. 10c, for three robots with varying speeds in the most cluttered environment. The size of the implicit partitions are proportional to the speeds of the robots. The centre was well covered since several parts are observed at once from these locations and therefore have high reward. The average sum of weighted nodes was 199.2 out of the maximum 221, with standard deviation 9.1, worst case 175 and best case 211. The average runtime was 17.6 s with a standard deviation

of 0.6 s. The performance was almost as consistent in this more complex scenario, and the solution paths have credible partitioning between multiple robots, selected lower cost locations within regions and favoured high-reward locations with overlapping viewpoint regions.

The fourth scenario trialled five robots in the high clutter environment, shown in Fig. 10d. The robots were given equal budgets such that it was just enough to be possible to collect 100 % of the rewards. In the Fig. 10d example trial, the robots have successfully shared the workload to find 5 approximately equal length paths that collectively visit all of the goal regions. Over the 100 trials, the average sum of weighted nodes was 218.8 out of the maximum 221, with standard deviation 3.6, worst case 206 and best case 221. Full coverage was achieved by 70 of the trials. For the trials that achieved suboptimal results, the viewpoint regions in the bottom left of the environment were more often missed since there is less incentive to visit that area. The average runtime was 22.1 s with a standard deviation of 0.6 s.

6.3 Comparison to Dec-MCTS

In these experiments, we investigate the benefits of planning over continuous space by comparing the proposed SOM planner to the recently proposed decentralised Monte Carlo tree search (Dec-MCTS) algorithm (Best et al., 2016a). Dec-MCTS is a decentralised planning algorithm that is applicable to general multi-robot problem formulations. In Best et al. (2016a), Dec-MCTS was demonstrated to perform well in two active perception problem formulation, one of which is closely related to the formulation addressed in this paper. There are several important differences between Dec-MCTS and our SOM approach. In particular, Dec-MCTS is decentralised, applicable to general objective functions and motion models, provides theoretical guarantees, and requires discretising the action space. While the SOM approach is centralised, is an efficient solution for a particular problem formulation, and effectively plans over continuous space. While these differences make it difficult to demonstrate a fair performance comparison, we show experiments here that highlight the benefit of planning over continuous space for our problem formulation (as in the SOM algorithm) rather than requiring a discretisation of the environment (as in Dec-MCTS).

These experiment were performed with 3 robots using the high-clutter dataset shown previously in Fig. 10 (c,d). In these experiments, the problem is discretised for Dec-MCTS using a probabilistic roadmap (PRM) with vertices $\mathcal{V}$ randomly placed in the viewpoint regions. Also, since Dec-MCTS requires a fixed start location, these experiments were performed using fixed start positions spread out near the centre of the environment, and the end positions are variables to be optimised by the planner. The experiments were performed with varying number of PRM vertices $\mathcal{V}$ for Dec-MCTS. Each scenario was repeated for 100 trials with this single problem instance. For each trial, a new set of PRM vertices $\mathcal{V}$ was randomly generated. Dec-MCTS was run until convergence was observed, which was between several seconds and several minutes depending on the size of $\mathcal{V}$. The SOM trials took 15 s each.

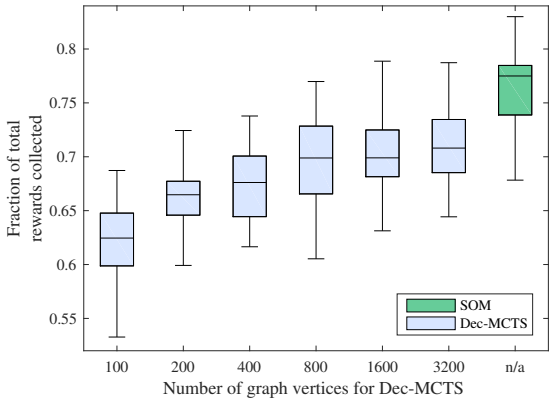


Fig. 11 Comparison between the proposed SOM approach and Dec-MCTS (Best et al., 2016a) with varying discretisation. For 3 robots in the high-clutter environment.

The results are shown in Fig. 11. The rewards collected by Dec-MCTS clearly improves when using a finer discretisation. This is because having more roadmap vertices $\mathcal{V}$ increases the probability of vertices being placed at valuable positions, e.g., positions that intersect multiple viewpoint regions and have relatively low travel-cost to other valuable vertices. On the other hand, the proposed SOM approach searches over the continuous space to adaptively find valuable positions for the path waypoints. This allowed the SOM approach to significantly outperform Dec-MCTS in all cases. Theoretically, Dec-MCTS would achieve the performance of SOM given a sufficient discretisation, but the computation and memory requirements would be intractable.

7 Online Exploration and Active Perception

In this section, we generalise the active perception scenario in the previous section to online scenarios for a team of robots. The robots make long-range 3D point-cloud observations to learn viewpoint regions and move to selected viewpoint regions to collect these rewards by observing the object parts at close range. The robots must plan to balance their workload between visiting the

currently known viewpoint regions and making observations to discover viewpoint goal regions. This is achieved by introducing exploration rewards as new viewpoint regions in unexplored areas of the environment. First, we formalise this observation model and planning scenario, then present results that illustrate the behaviour of the algorithm in online settings and highlight advantages of this formulation in comparison to short-horizon planning.

7.1 Online planning scenario

For these experiments, each robot has two 3D point-cloud sensing modalities such that high-quality observations are made at close range and coarse observations are made at long range. The close-range sensor is used to fulfil the primary perception task and has the same observation model as in Sec. 6.1. The long-range sensor is used to discover new objects and associated viewpoint regions and rewards. These two modalities could be provided by two separate sensors or by a single sensor where a close range is required to achieve a desired resolution. The viewpoint regions for the primary perception task are generated as described in Sec. 6.1 based on the point clouds that have been observed by the long-range sensor. Exploration is encouraged by introducing new viewpoint regions and rewards. This is achieved by placing a uniform grid of goals in the unexplored areas. The density of goals and their rewards can be selected to achieve a desired balance between exploration and exploitation. Each exploration goal has an associated circular viewpoint region with radius equal to the close sensing range and the rewards are uniformly set to 1. We use the close range rather than the long range for the exploration nodes since this results in a more accurate prediction of the travel distance required to observe discovered objects at close range. In these experiments, we use the above definition for exploration goals since it avoids making strong assumptions about the environment. However, if more prior knowledge were available, such as a belief of non-uniform density of objects, then more elaborate formulations could be used instead.

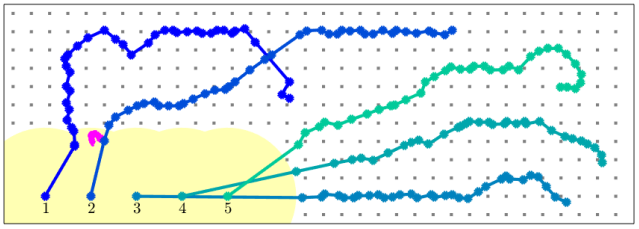
The simulations cycle between four phases: i) compute the viewpoint regions for both the primary perception task and exploration, ii) plan the paths for the team of robots with the SOM algorithm, iii) drive the robots a fixed distance along the planned paths, and iv) make new point-cloud observations with the long-range and short-range modalities. A team of five robots move through the 140×50 m environment shown in Fig. 12, which consists of the medium-, low- and high-clutter 3D point-cloud datasets (from Sec. 6) placed side-by-side from left to right in an enlarged environment. The long-range sensing range is 15 m and the close range is 4 m. These observations are simulated using the dataset by truncating the Velodyne measurements. All robots have an initial travel budget of 100 m travel distance, make observations at 1 m intervals along the path and replan after every 15 m. After each replanning phase, the remaining travel budget is reduced by 15 m. Each robot has a fixed start position and known current position for each replanning phase.

The planner uses an open-path graph topology with fixed start positions (as defined in Sec. 4.2). For the first planning round we set $\sigma_0 = 4$ and use arbitrary initial plans around the start positions. Online replanning is performed more effectively by adapting the previously planned paths and using $\sigma_0 = 2$. We use $\delta = 0.002$ for most experiments, and analyse the effect of these parameters in Sec. 7.2.4.

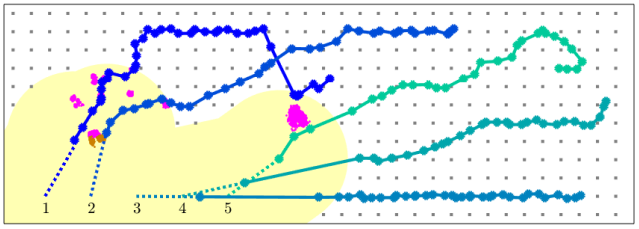
7.2 Results

The following results demonstrate: i) the algorithm achieves better performance when using a long planning horizon compared to a short horizon, ii) the effect of exploration reward density on performance, iii) the algorithm achieves comparable performance when planning online with partial information to when planning offline with full information, and iv) the algorithm efficiently adapts previous solutions when replanning.

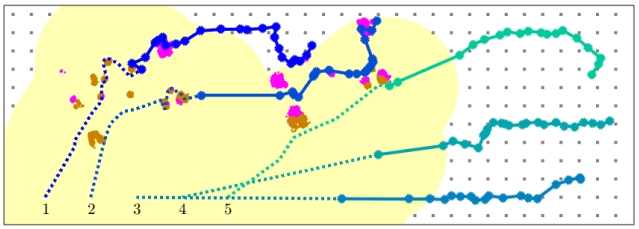
Fig. 12 illustrates an example of the behaviour of the algorithm when replanning. (a) Initially, only a single object has been observed from the start positions. Robots 1 and 2 cooperate by planning to observe both sides of this object at close range before proceeding to explore the top left of the environment. Robots 3, 4 and 5 evenly spread out to explore the right side of the environment. (b) After 15 m has been travelled by each robot, more objects are discovered by the long range sensor. Robots 1, 2 and 3 make minor refinements to their plans to make close-range observations of the new objects. (c) After 45 m, robot 3 discovers several more objects in the middle. Rather than robot 3 visiting these discovered goals itself, it instead decides to continue exploring to the right since robot 2 plans to visit these goals later. (d) A large number of objects are discovered on the right and robots 3, 4 and 5 cooperate to share these goals. (e) Once the budgets are exhausted, the robots have explored nearly all of the environment while also visiting 344 out of 420 close-range weighted goals. The robots successfully cooperated by rarely crossing paths or making duplicated observations.



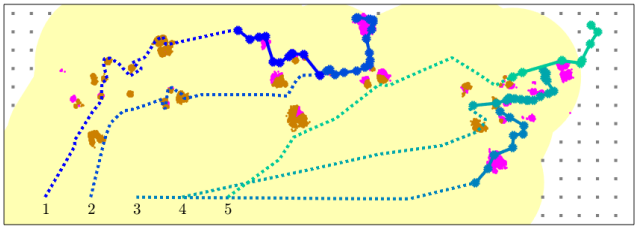
(a) Initial plans. Plan observes 254/330 weighted nodes.



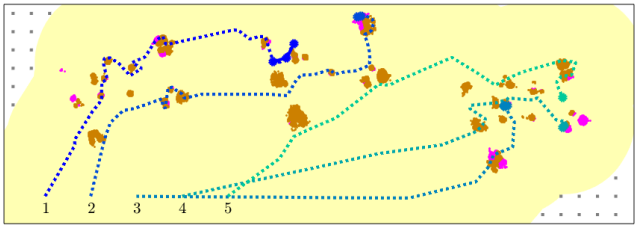
(b) 15 m travelled. Plan observes 274/362 weighted nodes.



(c) 45 m travelled. Plan observes 202/294 weighted nodes.

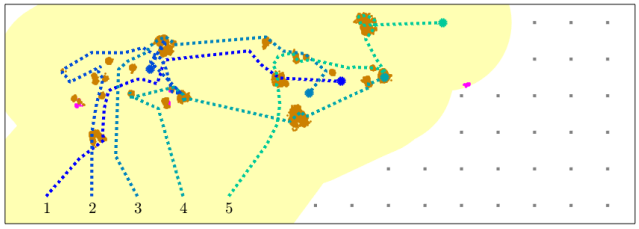


(d) 75 m travelled. Plan observes 183/262 weighted nodes.

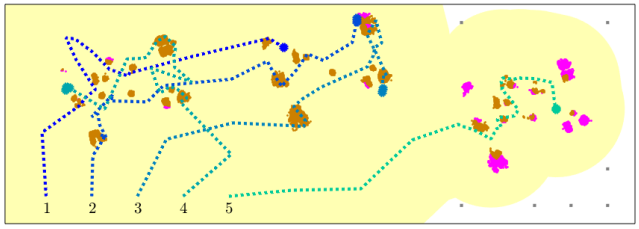


(e) Final executed paths (100 m). The robots observed 309/330 exploration goals and 344/420 primary nodes.

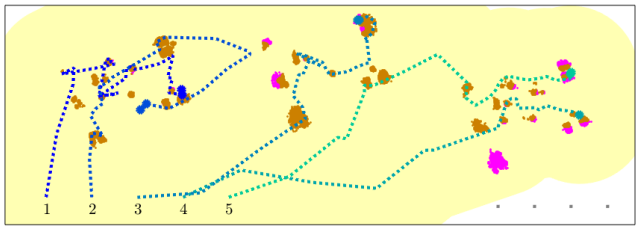
Fig. 12 Example run of the online experiments. Dotted lines are executed paths and solid lines are planned paths. Yellow regions have been explored. Point-clouds are observed at close-range (brown) and long-range (pink).



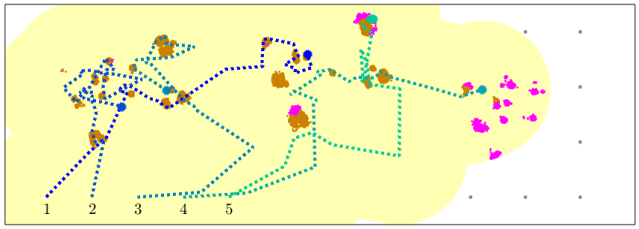
(a) 15 m planning horizon. Medium exploration density. The robots observed 267/420 primary nodes.



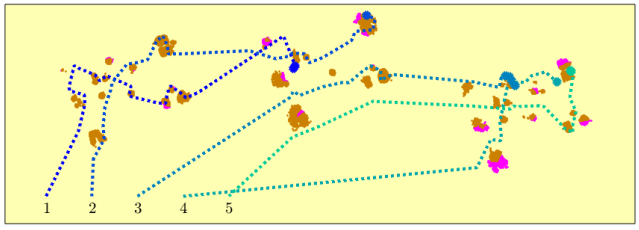
(b) 45 m planning horizon. Medium exploration density. The robots observed 303/420 primary nodes.



(c) Full planning horizon. Medium exploration density. The robots observed 332/420 primary nodes.



(d) Full planning horizon. Low exploration density. The robots observed 255/420 primary nodes.



(e) Full point-cloud is available offline. Full planning horizon. The robots observed 376/420 primary nodes.

Fig. 13 Example paths executed for (a-c) different planning horizons, (c-d) exploration reward densities and (e) offline full-observability.

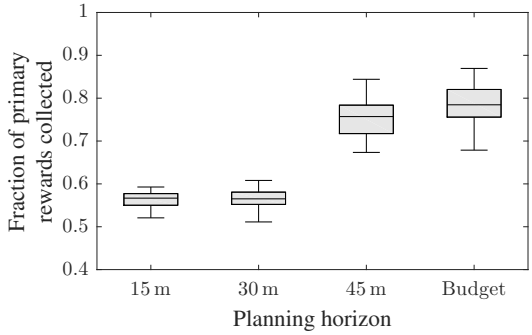


Fig. 14 Comparison of planners with different planning horizons. The 'budget' horizon optimises the entire remaining budget of each robot. Each scenario was performed 10 times.

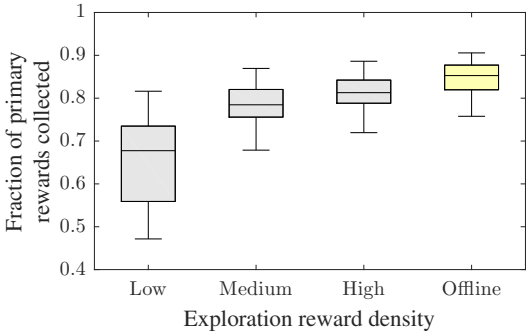


Fig. 15 Comparison of planning performance for different densities of exploration rewards. The full information scenario shows the performance of an offline planner with all primary viewpoint regions and rewards known in advance. Each scenario was performed 10 times.

7.2.1 Planning horizon

Fig. 14 compares online planning with the entire mission (100 m) as the planning horizon to when planning with shorter horizons. The longer planning horizons result in a significantly improved performance over the shorter horizons. The reason for this is illustrated in Fig. 13. For the shortest horizon (a), the objects on the left are discovered first and since these discovered goals cannot be satisfied by a single robot with a 15 m budget, the other robots also decided to visit these objects. As a result, the right side of the environment is never explored. As the mission progressed, the robots were left with no goals reachable within their budgets since they were already visited by other robots. Conversely, the longer planning horizons (b,c) enabled the robots to cooperate to explore the rest of the environment and visit the discovered objects. The longest planning horizon (c) achieved the best results since two robots managed to reach the dense group of objects on the right.

7.2.2 Exploration reward density

The algorithm generates paths that naturally balance between exploring the environment to discover new objects and visiting the objects at close-range to make high-quality observations. This balance can be influenced by selecting the density of exploration goals. If the density is low, as in Fig. 13d, then the robots have little incentive to visit unexplored regions and will instead focus on observing discovered objects. The robots in (d) did not perform well since they only just reached the objects on the right. However, if there were no objects in the right of the environment, then (d) would have outperformed the higher-density planners since it achieved better coverage of the discovered objects. This trend is also illustrated in Fig. 15: the higher exploration density scenarios outperformed the lower density scenarios in most trials for this environment. We note that a similar behaviour of balancing between exploring and exploiting occurs if the exploration goal rewards are varied rather than the density.

7.2.3 Partial information

For the scenario in Fig. 12e and the yellow column of Fig. 15, the planner had full knowledge of all of the goals offline and the algorithm is able to exploit this information to outperform the other scenarios. In Fig. 12e we see the robots do not need to spend their budget exploring empty space and instead take the most direct routes to their selected viewpoint regions. However, the partial-information scenarios still achieved reasonable results, despite not having access to this valuable information upfront. The high exploration density scenario collected on average 95% of the reward collected by the full-information scenario.

7.2.4 Adaptive replanning

We now analyse the benefits of adapting the previous solution when replanning compared to restarting the algorithm from the beginning. The results are shown in Fig. 16 for various combinations of parameter values. The parameter δ has the largest effect on computation time since this parameter directly influences the number of iterations before σ reaches the termination threshold; the $\delta = 0.002$ scenarios had an average runtime of 12 s during each replanning step and the $\delta = 0.004$ scenarios performed replanning more efficiently with a runtime of 3 s. The σ_0 parameter directly affects the ratio of the time spent making large global changes (when σ is large) to the time spent making smaller local refinements (when σ is small).

In all of the scenarios, reusing the previous solutions helped the algorithm perform replanning and made a

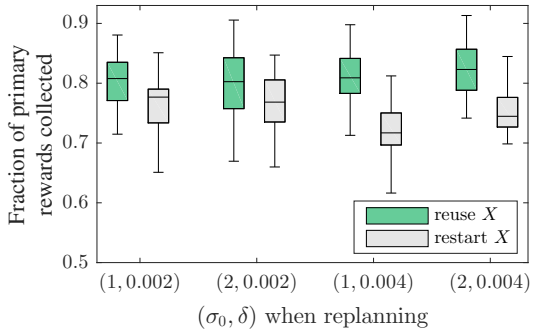


Fig. 16 Comparison between adapting the previous solution when replanning to clearing the path and starting again. Larger σ_0 parameter values result in more proportion of time is spent making global adaptations. Larger δ parameter values result in fewer epochs and faster planning time. Each scenario was performed 10 times.

statistically significant improvement to the collected rewards (t-test $p < 0.001$). There was no significant difference between the rewards collected for the different combinations of parameters when adapting the previous solution, even for cases with much fewer epochs ($\delta = 0.004$). When restarting the solution, the performance was poorer when the number of epochs was reduced, since it requires more iterations to adapt from the initial solution to a reasonable solution. There was a significant improvement ($p < 0.001$) for the ($\sigma_0 = 2, \delta = 0.004$) case over the ($\sigma_0 = 1, \delta = 0.004$) case when restarting the solution since the larger σ values result in more global adaptations for reaching an initial reasonable solution. Overall, these results highlight that the algorithm can effectively adapt previous solutions so that replanning can be performed more efficiently. This is particularly advantageous in online scenarios where the plans need to adapt to small changes in the objectives as the robots make observations.

8 Conclusion

We have proposed a new formulation and approach for multi-robot active perception problems. The objectives are defined as a set of continuous viewpoint regions, and the robots coordinate to maximise coverage of these regions. Self-organising maps is a fitting choice for developing solution algorithms; they can select favourable observation locations within continuous regions, while simultaneously optimising the full paths of the robots. Optimising the full paths, i.e., planning over a long time horizon, results in significant performance improvements over greedy and short-horizon planning. Our new SOM formulation addresses scenarios with non-uniform observation rewards, budget constraints, polygonal observation regions and multiple robots. The algorithm has polynomial time-complexity, converges towards a final solution, and is anytime. Additionally, we demonstrated that the formulation is suitable for online scenarios where the objectives change over time and the planner needs to efficiently adapt the plans to meet the new requirements. We also showed how the planner can be used to balance between exploring the environment to obtain new information and making high-quality observations of known objects. Our implementation achieved reasonable clock time performance of milliseconds to seconds. Overall, our results show that the proposed method enables multi-robot planning for budgeted active perception tasks with continuous sets of candidate viewpoints and multi-step planning horizons.

8.1 Future work

The formulation, approach and results motivate several avenues of future work. We discuss several ideas as follows: problem variants with different travel cost functions, variants to the sensor model formulation, decentralised extensions, and future hardware experiments.

The SOM algorithm is designed particularly for environments with Euclidean-distance costs. We are interested in extending the approach for scenarios with obstacles or non-holonomic constraints. Several ideas have been proposed for extending SOM algorithms for such scenarios, typically by combining an SOM approach with other planning algorithms, such as RRT (Faigl, 2016b). In non-holonomic scenarios, it would be an interesting challenge to incorporate orientation-dependent observations; a promising approach may be to approximate the problem in a high-dimensional Euclidean space (Kulich et al., 2016). Inter-robot collision avoidance is likely to be challenging to incorporate into an SOM algorithm due to the temporal constraints, but a decoupled approach could be an appropriate solution. However, we note that the SOM approach tends to find solutions where the robots' paths do not cross; therefore additional collision avoidance planning may not be necessary. It would also be interesting to design revised approaches for the most challenging instances of the considered problem; for example, instances with large variances in the spatial-density of nodes or robot budgets could require unusual robot partitioning that may not emerge from our current approach.

Our formulation is motivated by the fact that the performance of perception algorithms is sensitive to the choice of viewpoints. Viewpoint correlations can be conveniently expressed in our formulation such that all viewpoints within a region are considered correlated, and partial correlation can be expressed with overlapping regions. While this formulation is generally applicable,

it may also be convenient to express correlations by varying the rewards for each polygon, which may be addressed with a modification to the adaptation procedure (Faigl and Váňa, 2016). Other modifications to the reward function formulation could include extensions for teams of robots with heterogeneous sensing, which could readily be addressed by defining a different set of viewpoint regions and rewards for each robot. Other interesting problem generalisations include time-varying objectives for moving targets (Best et al., 2017; Hönig and Ayanian, 2016), and perception models with probabilistic viewpoint regions (Best et al., 2017; Best and Fitch, 2016). Also, while our experiments used a generic perception model to define the regions and rewards, this data processing can be adapted for the perception task at hand, such as by using other formulations for modelling 3D objects and predicting observations (Martens et al., 2017). A potential limitation for our algorithm is the runtime complexity dependence on the relative reward weights. In our recent work (Faigl, 2017) we avoid the node duplication by proposing an alternative adaptation function that results in smaller adaptations in each epoch, but is likely to be more efficient in cases where there is a large variance in rewards.

In many practical scenarios, such as farms and warehouses with permanent infrastructure, multi-robot coordination can be performed by a centralised server. However, in other scenarios it is necessary to decentralise the planning efforts and consider communication constraints, which presents new algorithmic and practical challenges (Best et al., 2016a; Corah and Michael, 2017; Kassir et al., 2015; Xu et al., 2013). A decentralised version of our SOM algorithm may be formulated by combining decentralised robot-node allocation with single-robot SOMs or small teams of multi-robot SOMs. These two components could interact in a similar way to Dec-MCTS (Best et al., 2016a) to optimise the joint-action space.

While we have performed extensive simulated experiments, in the future we would like to run our SOM approach onboard real robots in real-world scenarios. This would require addressing additional multi-robot challenges, such as dealing with decentralised data fusion, localisation uncertainty, and being robust to unreliable communication.

References

Angéniol B, de la C Vaubois G and Texier JYL (1988) Self-organizing feature maps and the travelling salesman problem. *Neural Networks* 1(4): 289–293.

Archetti C, Hertz A and Speranza MG (2007) Metaheuristics for the team orienteering problem. *J. Heuristics* 13(1): 49–76.

Atanasov N, Ny JL, Daniilidis K and Pappas GJ (2015) Decentralized active information acquisition: Theory and application to multi-robot SLAM. In: *Proc. of IEEE ICRA*. pp. 4775–4782.

Atanasov N, Sankaran B, Le Ny J, Pappas G and Daniilidis K (2014) Nonmyopic view planning for active object classification and pose estimation. *IEEE Trans. Robot.* 30(5): 1078–1090.

Bargoti S, Underwood JP, Nieto JI and Sukkarieh S (2015) A pipeline for trunk detection in trellis structured apple orchards. *J. Field Robot.* 32(8): 1075–1094.

Becerra I, Valentín-Coronado LM, Murrieta-Cid R and Latombe JC (2016) Reliable confirmation of an object identity by a mobile robot: A mixed appearance/localization-driven motion approach. *Int. J. Robot. Res.* 35(10): 1207–1233.

Bektas T (2006) The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega* 34(3): 209–219.

Best G, Cliff O, Patten T, Mettu R and Fitch R (2016a) Decentralised Monte Carlo tree search for active perception. In: *Proc. of WAFR*.

Best G, Faigl J and Fitch R (2016b) Multi-robot path planning for budgeted active perception with self-organising maps. In: *Proc. of IEEE/RSJ IROS*. pp. 3164–3171.

Best G and Fitch R (2016) Probabilistic maximum set cover with path constraints for informative path planning. In: *Proc. of ARAA ACRA*.

Best G, Martens W and Fitch R (2017) Path planning with spatiotemporal optimal stopping for stochastic mission monitoring. *IEEE Trans. Robot.* 33(3): 629–646.

Binney J and Sukhatme G (2012) Branch and bound for informative path planning. In: *Proc. of IEEE ICRA*. pp. 2147–2154.

Bircher A, Kamel M, Alexis K, Burri M, Oettershagen P, Omari S, Mantel T and Siegwart R (2016) Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots. *Auton. Robots* 40(6): 1059–1078.

Bourgault F, Makarenko A, Williams S, Grocholsky B and Durrant-Whyte H (2002) Information based adaptive robotic exploration. In: *Proc. of IEEE/RSJ IROS*. pp. 540–545.

Cao N, Low KH and Dolan JM (2013) Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In: *Proc. of AAMAS*. pp. 7–14.

Charrow B (2015) *Information-theoretic active perception for multi-robot teams*. PhD Thesis, University of Pennsylvania.

Chekuri C and Pal M (2005) A recursive greedy algorithm for walks in directed graphs. In: *Proc. of IEEE FOCS*. pp. 245–253.

Chen S, Li Y and Kwok NM (2011) Active vision in robotic systems: A survey of recent developments. *Int. J. Robot. Res.* 30(11): 1343–1377.

Cochrane EM and Beasley JE (2003) The co-adaptive neural network approach to the euclidean travelling salesman problem. *Neural Networks* 16(10): 1499 – 1525.

Corah M and Michael N (2017) Efficient online multi-robot exploration via distributed sequential greedy assignment. In: *Proc. of Robotics: Science and Systems*.

Cunningham-Nelson S, Moghadam P, Roberts J and Elfes A (2015) Coverage-based next best view selection. In: *Proc. of ARAA ACRA*.

Dang DC, El-Hajj R and Moukrim A (2013a) A branch-and-cut algorithm for solving the team orienteering problem. In: *Proc. of CPAIOR*. Springer, pp. 332–339.

Dang DC, Guibadj RN and Moukrim A (2013b) An effective PSO-inspired algorithm for the team orienteering problem. *Eur. J. Oper. Res.* 229(2): 332–344.

Dornhege C, Kleiner A, Hertle A and Kolling A (2016) Multirobot coverage search in three dimensions. *J. Field Robot.* 33(4): 537–558.

Faigl J (2010) Approximate solution of the multiple watchman routes problem with restricted visibility range. *IEEE Trans. Neural Networks* 21(10): 1668–1679.

Faigl J (2016a) An application of self-organizing map for multirobot multigoal path planning with minmax objective. *Comput. Intel. Neurosc.* DOI:10.1155/2016/2720630.

Faigl J (2016b) On self-organizing map and rapidly-exploring random graph in multi-goal planning. In: *Advances in Self-Organizing Maps and Learning Vector Quantization*. Springer, pp. 143–153.

Faigl J (2017) On self-organizing maps for orienteering problems. In: *Proc. of IJCNN*. pp. 2611–2620.

Faigl J and Hollinger G (2014) Unifying multi-goal path planning for autonomous data collection. In: *Proc. of IEEE/RSJ IROS*. pp. 2937–2942.

Faigl J and Hollinger GA (2017) Autonomous data collection using a self-organizing map. *IEEE Trans. Neur. Net. Lear.* doi:10.1109/TNNLS.2017.2678482.

Faigl J, Kulich M and Přeučil L (2012) Goal assignment using distance cost in multi-robot exploration. In: *Proc. of IEEE/RSJ IROS*. pp. 3741–3746.

Faigl J, Pěnička R and Best G (2016) Self-organizing map-based solution for the orienteering problem with neighborhoods. In: *Proc. of IEEE SMC*. pp. 1315–1321.

Faigl J and Váňa P (2016) Self-organizing map for data collection planning in persistent monitoring with spatial correlations. In: *Proc. of IEEE SMC*. pp. 3264–3269.

Galceran E and Carreras M (2013) A survey on coverage path planning for robotics. *Robot. Auton. Syst.* 61(12): 1258–1276.

Garg S and Ayanian N (2014) Persistent monitoring of stochastic spatio-temporal phenomena with a small team of robots. In: *Proc. of Robotics: Science and Systems*.

Gunawan A, Lau HC and Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. *Eur. J. Oper. Res.* 255(2): 315 – 332.

Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* 126(1).

Hollinger G, Singh S, Djugash J and Kehagias A (2009) Efficient multi-robot search for a moving target. *Int. J. Robot. Res.* 28(2): 201–219.

Hollinger GA, Mitra U and Sukhatme GS (2011) Active classification: Theory and application to underwater inspection. In: *Proc. of ISRR*.

Hönig W and Ayanian N (2016) Dynamic multi-target coverage with robotic cameras. In: *Proc. of IEEE/RSJ IROS*. pp. 1871–1878.

Kassir A, Fitch R and Sukkarieh S (2015) Communication-aware information gathering with dynamic information flow. *Int. J. Robot. Res.* 34(2): 173–200.

Kriegel S, Brucker M, Marton ZC, Bodenmuller T and Suppa M (2013) Combining object modeling and recognition for active scene exploration. In: *Proc. of IEEE/RSJ IROS*. pp. 2384–2391.

Kulich M, Faigl J and Přeučil L (2011) On distance utility in the exploration task. In: *Proc. of IEEE ICRA*. pp. 4455–4460.

Kulich M, Sushkov R and Přeučil L (2016) Speed-up of self-organizing networks for routing problems in a polygonal domain. In: *Proc. of IEEE/RSJ IROS 10th International Workshop on Cognitive Robotics*.

Lagoudakis MG, Markakis E, Kempe D, Keskinocak P, Kleywegt AJ, Koenig S, Tovey CA, Meyerson A and Jain S (2005) Auction-based multi-robot routing. In: *Proc. of Robotics: Science and Systems*.

Likhachev M, Ferguson DI, Gordon GJ, Stentz A and Thrun S (2005) Anytime dynamic A*: An anytime, replanning algorithm. In: *Proc. of ICAPS*. pp. 262–

271.

Martens W, Poffet Y, Soria PR, Fitch R and Sukkarieh S (2017) Geometric priors for Gaussian process implicit surfaces. *IEEE Robot. Autom. Lett.* 2(2): 373–380.

Mathew N, Smith S and Waslander S (2013) A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In: *Proc. of IEEE ICRA*. pp. 3497–3502.

McMahon J and Plaku E (2017) Autonomous data collection with limited time for underwater vehicles. *IEEE Robot. Autom. Lett.* 2(1): 112–119.

Noon CE and Bean JC (1989) An efficient transformation of the generalized traveling salesman problem. Technical Report 89-36, Department of Industrial and Operations Engineering, University of Michigan.

Patten T (2017) *Active Object Classification from 3D Range Data with Mobile Robots*. PhD Thesis, The University of Sydney.

Patten T, Kassir A, Martens W, Douillard B, Fitch R and Sukkarieh S (2015) A Bayesian approach for time-constrained 3D outdoor object recognition. In: *Proc. of IEEE ICRA Workshop on Scaling Up Active Perception*.

Patten T, Martens W and Fitch R (2017) Monte Carlo planning for active object classification. *Autonomous Robots* doi:10.1007/s10514-017-9626-0.

Patten T, Zillich M, Fitch R, Vincze M and Sukkarieh S (2016) Viewpoint evaluation for online 3-D active object classification. *IEEE Robot. Autom. Lett.* 1(1): 73–81.

Peng C, Roy P, Luby J and Isler V (2016) Semantic mapping of orchards. *IFAC-PapersOnLine* 49(16): 85–89.

Quattrini Li A, Cipolleschi R, Giusto M and Amigoni F (2016) A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. *Auton. Robots* 40(4): 581–597.

Robin C and Lacroix S (2015) Multi-robot target detection and tracking: Taxonomy and survey. *Auton. Robots* 40(4): 729–760.

Singh A, Krause A, Guestrin C and Kaiser WJ (2009) Efficient informative sensing using multiple robots. *J. Artif. Intel. Res.* 34(2): 707.

Smith SL and Imeson F (2017) GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Comput. Oper. Res.* 87: 1–19.

Somhom S, Modares A and Enkawa T (1997) A self-organising model for the travelling salesman problem. *J. Oper. Res. Soc.* 48(9): 919–928.

Somhom S, Modares A and Enkawa T (1999) Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Comput. Oper. Res.* 26(4): 395–407.

Toth P and Vigo D (2001) *The vehicle routing problem*. SIAM.

Tucci M and Raugi M (2010) Stability analysis of self-organizing maps and vector quantization algorithms. In: *Proc. of IJCNN*. pp. 1–5.

van Hoof H, Kroemer O and Peters J (2014) Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Trans. Robot.* 30(5): 1198–1209.

Vansteenwegen P, Souffriau W and Oudheusden DV (2011) The orienteering problem: A survey. *Eur. J. Oper. Res.* 209(1): 1–10.

Wohlkinger W, Aldoma A, Rusu RB and Vincze M (2012) 3DNet: Large-scale object class recognition from CAD models. In: *Proc. of IEEE ICRA*. pp. 5384–5391.

Wohlkinger W and Vincze M (2011) Ensemble of shape functions for 3D object classification. In: *Proc. of IEEE ROBIO*. pp. 2987–2992.

Wu K, Ranasigne R and Dissanayake G (2015) Active recognition and pose estimation of household objects in clutter. In: *Proc. of IEEE ICRA*. pp. 4230–4237.

Xu Z, Fitch R, Underwood J and Sukkarieh S (2013) Decentralized coordinated tracking with mixed discrete-continuous decisions. *J. Field Robot.* 30(5): 717–740.

Yu J, Schwager M and Rus D (2016) Correlated orienteering problem and its application to persistent monitoring tasks. *IEEE Trans. Robot.* 32(5): 1106–1118.

Zhang WD, Bai YP and Hu HP (2006) The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP. *Appl. Math. Comput.* 172(1): 603 – 623.

Zlot R, Stentz A, Dias M and Thayer S (2002) Multi-robot exploration controlled by a market economy. In: *Proc. of IEEE ICRA*, volume 3. pp. 3016–3023.

A Convergence of SOM

In this appendix, we elaborate on the convergence properties discussed in Remark 2 of Sec. 5.1.1 to provide insight into the behaviour of the algorithm.

There are two key phases of the algorithm to consider when analysing the convergence properties. After epoch $i_{max} = 1/\delta$, the neighbourhood function definition (1) ensures that no further adaptations occur, and thus the algorithm is guaranteed to have converged (Lemma 2). However, the algorithm will typically converge prior to i_{max}, but after the neighbourhood function (1) reaches 0 for $l > 0$. Because such a convergence depends on the spatial configuration of the viewpoint regions, it is not

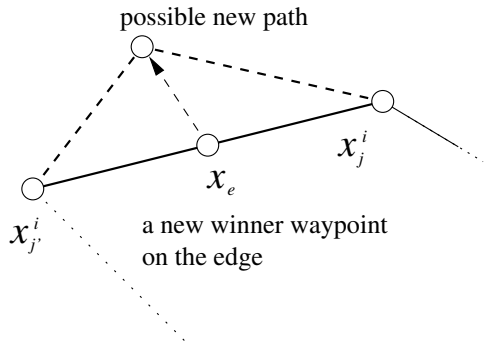


Fig. 17 Illustration of a scenario where the winner is on an edge of the path. All possible adaptations result in an increased path length.

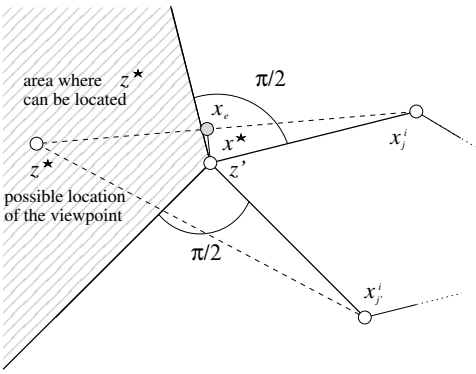


Fig. 18 Illustration of a scenario where the winner is an existing waypoint. The path after the adaptation to $z^\star$ is shown as the dashed lines connecting x_j^i with $z^\star$ and $z^\star$ with $x_{j'}^i$. This adaptation results in an increased path length.

possible to prove convergence prior to i_{max} for all possible cases. In fact, it is possible to show that for certain configurations, a particular waypoint of the network may oscillate between viewpoint locations during the learning epochs. However, these are specific cases and do not occur frequently; therefore, faster convergence occurs with a high probability. Furthermore, these oscillations do not occur to the *best* found solution, which is what is actually returned by the algorithm. In the remainder of this appendix, we discuss the intuition behind these claims of convergence. For simplicity and without loss of generality, we suppose a single robot problem ($R = 1$) and the viewpoint regions are defined as discrete points.

The crucial property for determining convergence is that, after some epoch, only the winner is moved towards the viewpoint location while the neighbours of the winner are not adapted. This occurs when $f(\sigma, l) = 0$ for $l > 0$, and thus only the waypoints with the cardinal distance $l = 0$ to $x^\star$ can be adapted, i.e., only the winner waypoint $x^\star$ is moved. In the case of $\sigma_0 = 4$ and $\delta = 0.002$, this occurs at learning epoch $i = 68$; this agrees with our empirical analysis of the convergence of solutions in Fig. 6.

The main intuition behind the convergence is related to the limited travel cost budget b^1. The adaptation is performed only if the sequence of waypoints satisfies the budget constraint after the adaptation. Let the value of the neighbourhood function be non-zero only for the winner waypoint, as described above. In cases where the winner is a new waypoint x_e on an edge (Alg. 2 line 13), the length of the path (sequence of waypoints X^i) must increase, as illustrated in Fig. 17. Thus, if edge nodes are selected as winners, the network will stop adapting once the budget limit is reached.

The other case to consider is when the winner is an existing waypoint $x^\star$. Let the neighbouring waypoints to $x^\star$ be x_j^i and $x_{j'}^i$. This scenario is illustrated in Fig. 18. The shaded area represents possible locations for $z^\star$ that would result in the existing waypoint $x^\star$ being selected as the winner. This area is defined as the intersection of the half-planes with boundaries perpendicular to $(x_j^i, x^\star)$ and $(x^\star, x_{j'}^i)$. In the Fig. 18 adaptation scenario, the length of the path increases in a similar way to the edge waypoint x_e winner case. Thus, this type of adaptation can only occur up until the cost budget is met.

However, when an existing $x^\star$ waypoint is selected as the winner, it is possible for the length of the path to decrease as a result of the adaptation. A simple example of such a situation is visualised in Fig. 19 with two fixed waypoints at z_1 and z_2. Assume the travel cost budget is such that the path can visit either z_3 or z_4, but it is impossible to visit both z_3 and z_4 without exceeding the budget. If the network is in the configuration shown in Fig. 19a, and the random permutation of the viewpoints be such that the node z_3 is presented as the first node, then the winner waypoint at z_4 will be adapted to z_3. When z_4 is presented, the network adapts back to z_4 (Fig. 19b). In this configuration, the waypoint may continue to oscillate between z_4 and z_3 until $f(\sigma, 0)$ finally reaches zero. However, it is rare for such a configuration of viewpoint locations to occur. Also, the network may also adapt towards other viewpoints, which will increase the path length until the cost budget is reached, causing the oscillations to eventually cease.

We also note that the algorithm maintains the *best* found solution (Alg. 1 line 15). When the network is oscillating as described above, the best found solution is likely to remain constant. This is the solution that is returned by the algorithm. An empirical verification of the presented intuition behind the solution convergence is reported in Fig. 5, where the solution does not change after 50 learning epochs. The network itself does not change after 70 epochs, which further supports the pre-

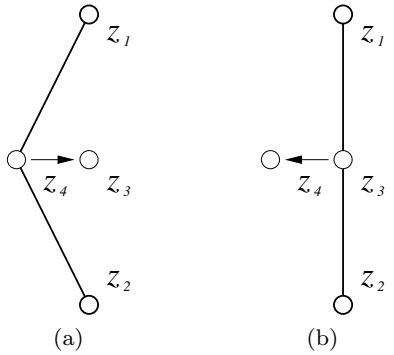


Fig. 19 An example scenario that may cause the network to oscillate between two viewpoint locations z_3 and z_4.

sented idea that the network typically converges much sooner than $i_{max} = 1/\delta$ learning epochs.