Placing and scheduling many depth sensors for wide coverage and efficient mapping in versatile legged robots

The International Journal of Robotics Research
XX(X):1–35

DOI: 10.1177/0278364919891776
journals.sagepub.com/home/ijr
SAGE

Martim Brandão[1], Rui Figueiredo[2], Kazuki Takagi[3], Alexandre Bernardino[2], Kenji Hashimoto[4], Atsuo Takanishi[3]

Abstract

This paper tackles the problem of designing 3D perception systems for robots with high visual requirements, such as versatile legged robots capable of different locomotion styles. In order to guarantee high visual coverage in varied conditions (e.g. biped walking, quadruped walking, ladder climbing), such robots need to be equipped with a large number of sensors, while at the same time managing the computational requirements that arise from such a system. We tackle this problem at both levels: sensor placement (how many sensors to install on the robot and where) and run time acquisition scheduling under computational constraints (not all sensors can be acquired and processed at the same time).

Our first contribution is a methodology for designing perception systems with a large number of depth sensors scattered throughout the links of a robot, using multi-objective optimization for optimal trade-offs between visual coverage and the number of sensors. We estimate the Pareto-front of these objectives through evolutionary optimization, and implement a solution on a real legged robot. Our formulation includes constraints on task-specific coverage and design symmetry, which lead to reliable coverage and fast convergence of the optimization problem. Our second contribution is an algorithm for lowering the computational burden of mapping with such a high number of sensors, formulated as an information-maximization problem with several sampling techniques for speed.

Our final system uses 20 depth sensors scattered throughout the robot, which can either be acquired simultaneously or optimally scheduled for low CPU usage while maximizing mapping quality. We show that, when compared to state-of-the-art robotic platforms, our system has higher coverage across a higher number of tasks, thus being suitable for challenging environments and versatile robots. We also demonstrate that our scheduling algorithm allows to obtain higher mapping performance than naive and state-of-the-art methods by leveraging on measures of information gain and self-occlusion at low computational costs.

Keywords

Design optimization, sensor placement, visual coverage, next-best-view, sensor scheduling, 3D mapping.

Introduction

The performance of robot locomotion can depend highly on the robot's visual coverage and the quality of the map used for navigation and planning. Real-world environments may be of challenging geometry and visual occlusions, which creates a demand for frequent, complete and precise visual measurements. Versatile legged-robots further complicate the perception problem, in that performance should be robust to locomotion mode and potentially very different robot configurations (e.g. crawling, walking, standing manipulation, ladder-climbing).

In this paper we introduce methods to guarantee wide visual coverage and high-quality 3D reconstruction on versatile-locomotion legged robots. One of the motivations

[1]Oxford Robotics Institute, University of Oxford, UK
[2]Instituto Superior Tecnico, Universidade de Lisboa, PT
[3]Waseda University, JP
[4]Meiji University, JP

Corresponding author:
Martim Brandão, Oxford Robotics Instiute, 23 Banbury Road, Oxford, OX2 6NN, UK.
Email: martim@robots.ox.ac.uk

for this work comes from the observation that most state-of-the-art robotic platforms use a single laser rangefinder or a few stereo cameras to deal with all of the robot's tasks. There are several problems with such an approach. First, while laser rangefinders do allow for large fields-of-view, they are slow to perceive in all directions Fallon et al. (2015). The limitations are clear in state-of-the-art robot platforms such as the ones who participated in the DARPA Robotics Challenge, which had slow update rates and low visual coverage as reported by some of the teams Fallon et al. (2015); Haynes et al. (2017). Even assuming the problem can be mitigated in the future by improved sensors and rotation speeds, robots with single head setups will be severely constrained in terms of self-occlusions and blind-spots. Such limitations could be fatal in robots capable of diverse walking styles, where contact locations and locomotion directions are arbitrary. In this paper we propose to instead use a large set of depth sensors scattered throughout a robot's links.

More concretely, the problem we are trying to solve in this paper is the following. We are given a budget to buy and install multiple depth sensors on a versatile legged robot. We want to decide how many of such sensors to buy, where to physically place them (in terms of the link to attach to and the pose of the sensors), and how to schedule their use such as to obtain high quality mapping (in terms of occupancy grid entropy). Our objectives are to obtain a good trade-off between mapping quality and the number of sensors used. Our solution here is to decouple the problem into: 1) Multi-objective optimization-based sensor placement design for high coverage and low sensor count; and 2) A sensor scheduling algorithm for high quality mapping given the sensor placement design of 1).

Our problem is important because the safety and efficiency of planning and control algorithms, as well as teleoperation interfaces, rely heavily on map quality and completeness. The problems are also challenging and interesting because

A. the robots we consider are capable of locomotion in varied styles (e.g. crawling, walking, ladder-climbing, etc) and as such the perception system should be designed to be reliable on all of those styles,

B. the sensors are in general not required to be placed on the body or a head, but could in principle be placed in any link, in a way that is optimal for visual coverage taking robot motion capabilities into account,

C. robots should ideally be self-contained, and so the computing resources dedicated to each sensor should be managed in smart and efficient ways so that all processing can be done on-board.

This paper introduces and evaluates algorithms that consider all such factors when designing a fully functioning perception system for versatile legged robots.

Regarding the design of the many-camera perception system, our approach to sensor placement is similar to that used in sensor networks Bodor et al. (2005); Hörster and Lienhart (2006); Chakrabarty et al. (2002): we use optimization to select and rotate sensors from a pre-defined set of possible locations such as to maximize coverage. An important difference with respect to the sensor networks literature in this paper is that sensors are not static but will move with respect to each other over time, according to the motion of the robots' joints. Our approach is to explicitly model the distribution of robot motion through motion exemplars, and estimate visual coverage over the different motions in expectation. In addition to that, and since we consider multi-contact locomotion tasks, we use task-specific optimization constraints which enforce that all planned contact points are visible from at least one sensor. And finally, we solve a multi-objective optimization problem instead of single-objective (coverage) optimization. Specifically we consider both the maximization of coverage and minimization of the number of sensors, since the latter is crucially related to real-world costs of installation, weight, maintenance and others. We estimate the Pareto-front of the multiple objectives, thus obtaining a set of solutions with optimal trade-offs. Through this approach, we can then use other considerations which are more difficult to encode (e.g. space for cabling, difficulty of attachment in particular locations) to actually choose the final design without compromising on main-objective trade-off optimality.

Our final design consists of 20 depth cameras scattered throughout the body, legs and feet of the legged robot WAREC-1 Hashimoto et al. (2017). The robot, shown in Figure 1, is targeted at rescue missions in challenging environments and is not only capable of multiple locomotion styles (item A above) but it is also approximately symmetric in all planes. This property increases versatility and efficiency since locomotion can be made in all directions (e.g. no need to rotate the body to look "forward" before climbing a ladder). It also motivates us to include symmetry as a constraint in the design optimization problem, leading to faster convergence speeds as we show in the section "Results".

The high visual coverage we obtain by using 20 depth cameras comes at the cost of additional processing needs, since for example point cloud filters need to be run for all cameras. We smartly schedule camera processing according

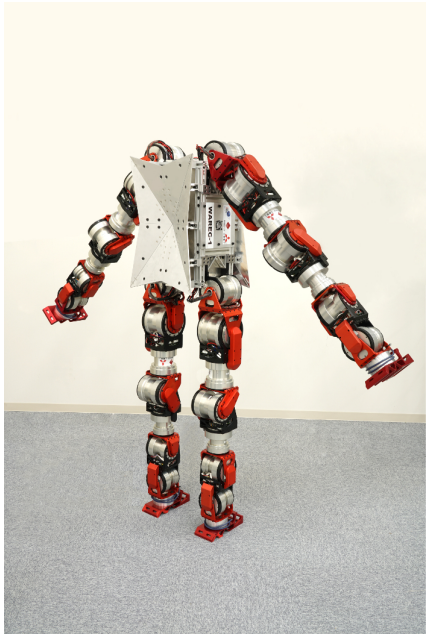


Figure 1. WAREC-1, our robotic platform (before attaching any visual sensors).

to predicted improvements in the environment mapping and available resource limitations using information gain metrics Delmerico et al. (2018). Note that even if scheduling reduces the speed at which different regions of space are processed, at least sensor acquiring order can be smartly controlled to focus on important viewing directions (e.g. locomotion direction, non-occluded cameras, etc), a kind of flexibility that laser rangefinders do not allow. In order to formulate our scheduling algorithm, we first pose the 3D volumetric mapping problem as a process of decision making under uncertainty, aimed at maximizing reconstruction quality. We leverage on the anticipated measurement qualities, which are provided by a probabilistic sensor error model, to develop an improved reward signal for Next Best View (NBV) planning based on expected entropy losses. Finally, we incorporate efficient sampling-based techniques that allow for efficient planning in a subset of possible perceptual actions, at the expense of minor decrease in reconstruction performance.

As a summary, the contributions of this paper are the following:

- We propose to optimize the number, position and orientation of a large set of depth sensors on a robot using a genetic algorithm for Pareto-front estimation, thus obtaining an optimal trade-off between visual coverage and the number of sensors
- We show that our method increases spherical coverage considerably when compared to common design approaches, and obtains good empirical coverage in terms of both wideness of coverage and contact point visibility
- We apply Next Best View (NBV) algorithms to the problem of sensor scheduling in many-sensor legged robots. While these algorithms are usually used to choose the next camera view point, here they are used to decide which of a set of sensors to acquire, given the current robot posture and map.
- We propose and evaluate the use of Upper Confidence Bounds (UCBs), pixel and camera sampling, self-occlusion information, and task-relevant priors for fast and highly informative scheduling reward signals.
- We demonstrate that our resource-constrained scheduling algorithm outperforms several scheduling algorithms and active vision baselines, both in terms of time to perceive new objects and 3D reconstruction quality.

The structure of the paper is the following: in the next section "Related work" we situate and contrast our paper with literature across multiple related fields. After that, we dedicate a section to sensor placement "Designing a many-sensor perception system" and one to scheduling "Scheduling and mapping on a many-sensor perception system". Our "Results" section proceeds similarly by first evaluating the sensor placement algorithm and then the scheduling algorithm. Our evaluation includes both simulation and real-robot experiments. We conclude and discuss our results, limitations and future work in the final "Conclusion" section.

Related work

Legged robot 3D perception-system designs

Most state-of-the-art robotic platforms use some combination of LIDAR and stereo for 3D reconstruction Fallon et al. (2015); Atkeson et al. (2015); Kaneko et al. (2015); Yoshiike et al. (2017); Tsagarakis et al. (2017); Radford et al. (2015); Karumanchi et al. (2017); Haynes et al. (2017); Hutter et al. (2017), as well as task-specific cameras for teleoperation Atkeson et al. (2015); Fallon et al. (2015); Kaneko et al. (2015); Yoshiike et al. (2017); Radford et al. (2015). Even though the literature describing such platforms does not quantitatively evaluate visual coverage or how effective certain sensor locations are, some discuss lack of visual coverage as a problem in operation Fallon et al. (2015); Haynes et al. (2017) and the need for scattering more

visual sensors throughout the robot as a solution Atkeson et al. (2015).

Teams in the DARPA Robotics Challenge (DRC) using the Atlas robot from Boston Dynamics relied on LIDAR and stereo Fallon et al. (2015); Atkeson et al. (2015), as well as fisheye cameras Fallon et al. (2015) or wrist and knee cameras Atkeson et al. (2015) for visual perception and teleoperation. The authors of Fallon et al. (2015) complain both about low visual coverage of the system and the slow 1Hz update rate of LIDAR. The CHIMP robot Haynes et al. (2017) employs a head setup with one LIDAR on each side (i.e. at the "ears") and forward-facing stereo. The authors also mention the system's occlusions were too high, in particular for accurate visual odometry. Also at DRC, the robot HRP-2Kai used a head design and LIDAR placement such as to be able to cover points at the robot's feet, as well as cameras on the hands' palms for manipulation tasks Kaneko et al. (2015). The robot E2-DR Yoshiike et al. (2017) has similar cameras on the hands, and a head-system similar to CHIMP's with two LIDAR sensors rotating along the yaw axis (one sensor at each ear) and a front-looking stereo and time-of-flight (TOF) camera. The quadruped legged robot ANYmal Hutter et al. (2017) also uses a pair of LIDAR, one in the front and one in the back of the body, as well as a pan-tilt sensor head on top of the body for specialized sensors. Robosimian Karumanchi et al. (2017) uses more sensors than most legged robots, attached to different areas of the trunk. It uses downward-looking stereo cameras on all sides, the belly, and the head. Valkyrie Radford et al. (2015) is perhaps the most sensing-intensive legged (humanoid) robot at present. It uses stereo on the head, TOF and LIDAR on the head, wrists and legs, and finally sonar on the waist. The authors do not describe how to efficiently manage such a large amount of data streams, nor do they quantitatively evaluate the sensor placement design.

In general, literature on state-of-the-art robotic platforms such as the ones just described, even those focused on providing large fields-of-view, has not precisely evaluated the effects of the design choices behind their perception systems. Furthermore, the problem of how to effectively deal with the amount of data that grows with the number of sensors has been ignored and mostly been passed on to the teleoperator's decision-making process. Contrary to this trend, in this paper we focus on precisely these two points. We believe that there is a need to systematically evaluate basic perception system features such as visual coverage, and to develop algorithms to manage resources efficiently.

Design optimization

This paper is closely related to the literature on design optimization. In mobile robotics, the problem has been mostly approached in the "evolutionary robotics" and graphics literature, for example for evolving robot designs Sims (1994) and soft creature designs Cheney et al. (2013) for locomotion in simulation. Brodbeck et al. (2018) also applies similar methodologies to evolve real cubic-module robots. And Schulz et al. (2017) uses simulation, optimization and user interfaces in a loop to design origami-like robots.

Although design optimization has not, to the best of our knowledge, been put to practice in real legged robot platforms, the topic is of high importance in the industry. Particularly, the field of multi-disciplinary design optimization deals with optimization formulations of complex design problems, and has been applied to aircraft, bridge, car, turbine design and others Martins and Lambe (2013). In industrial robotics, design optimization has been used to find optimal dimensions of grippers Saravanan et al. (2009), gearboxes Pettersson and Ivander (2009) and arm geometry Park and Asada (1993). In this paper we take a similar approach to the design of a perception system, in that we formulate it as an optimization problem with a design parameterized by sensor locations and rotations.

One issue with optimization-based design is that of defining a single objective function encoding all relevant factors (e.g. coverage, number of sensors, financial cost, complexity, time to build the system). Sometimes, no clear constraints exist on the different factors, but the best possible trade-off is wished for. Recent evolutionary algorithms can estimate Pareto fronts of multi-objective problems Deb et al. (2002); Zitzler et al. (2001), which is an effective way to lower the burden of formulation of design problems. For example, Saravanan et al. (2009) uses Pareto-front estimation to obtain trade-offs of different objectives for design optimization of robot grippers. The approach produces a set of solutions with optimal trade-offs, thus freeing the designer to pick solutions using other, more intuitive or difficult-to-encode considerations without fears of losing optimality on the main objectives. In this paper we use a similar approach, and compute the Pareto-front of visual coverage and the number of sensors during design optimization of our perception system. We then pick a design from the set of optimal solutions by using considerations not automatically extractable from robot CAD data, such as difficulty of sensor attachment, space for cabling, etc.

Visual coverage and sensor placement

Estimating sensor coverage, the focus of our design optimization method, is a problem which is present in many applications. Those range from sensor networks and video-surveillance to ergonomic design, robotic inspection, robot-based 3D object scanning, computer games and others. Coverage can be defined as the number (or percentage) of points of a region of interest which are within a sensor's feasible measurement range. For example, Mayol-Cuevas et al. (2009) quantifies where best to place a wearable camera on a human body by computing the percentage of visible directions on the sphere and on a region of interest using raycasts from many points placed throughout the body. Interestingly, the head is one of the optimal locations for a sensor in terms of visual coverage, implying humans are already optimal in sensor placement. Similar metrics are used in video-surveillance applications, for example by optimizing the visibility of paths of objects of interest on cameras Bodor et al. (2005), or the visibility of a set of points sampled from an importance distribution Hörster and Lienhart (2006). In inspection applications, the objective may be for a moving sensor to cover points at an object's boundary. For example Englot and Hover (2013) plans underwater robot trajectories that cover all points at an object's boundary.

Another definition of visual coverage is the average length of rays sent from a sensor (i.e. average distance to the first object hit) on uniformly sampled directions. The definition makes sense for sensors whose performance worsens with distance, and is used in tactical path-finding for non-player-characters in computer games. The objective there is, for example, to find locations where players can easily be targeted by "snipers" or to plan paths through locations good for "cover" (i.e. low visibility to avoid attacks) Millington and Funge (2016). Computer games also employ efficient representations for fast approximations of raycast-based visibility, such as depth-buffer cube maps van der Leeuw (2009).

In this paper we use the former definition of coverage: as the percentage of visible points on a set. In particular, one of the objective functions of our optimization algorithm is the percentage of points which is visible by at least one sensor, and points lie on a sphere which is centered at the base link of the robot and has a large radius. The objective is to allow the robot to see in all directions, which is especially important for versatile locomotion robots which can walk, crawl and climb in all directions. We also consider additional coverage constraints that guarantee complete visibility of task-specific points: in particular that the set of end-effector positions in motion are visible from the preceding stance by at least one sensor. We do this to allow for re-planning of contact positions and visual servoing applications.

The problem of choosing where to place sensors in order to optimize coverage is an important one in sensor networks and video-surveillance Bodor et al. (2005); Hörster and Lienhart (2006); Chakrabarty et al. (2002), where it is called "sensor placement". The goal there is to place sensors statically in the environment such as to be able to track people and other objects of interest for large distances and across sensors. For example, Horster formulates the problem as a binary integer program that maximizes coverage of a sampled blueprint by selecting which sensors (of different field-of-view properties) to place in which pre-selected points. Sensor placement has also been studied in mobile robotics, although to the best of our knowledge it has not been used within an optimization problem. For example, Keyes et al. (2006) evaluates two different options for sensor placement in terms of situation awareness for robot teleoperators. Mutlu et al. (2015) also evaluates different options for camera placements, but with the goal of reducing image blur on a snake robot.

In this paper we take a systematic optimization-based approach to the sensor placement problem as in video-surveillance Hörster and Lienhart (2006). However, our problem is more general and complex, since sensors placed on a legged articulated robot will move in position and orientation as a function of time, not in fixed patterns but freely within joint limits as a result of full-body trajectory planning Brandao et al. (2016). The generalized problem is then of optimizing coverage over the distribution of robot motions, which we approximate by a set of exemplar motions of different locomotion styles. Furthermore, visibility will be a non-convex function of sensor orientation in our case due to self-occlusions by the robot's links, which makes it difficult to find accurate approximate solutions. As a side note, the name "sensor placement" is also used in part of the literature to refer to the problem of planning a trajectory for a camera, which is then tracked by a robot by a subsequent planning stage Chen and Li (2004). This paper does not deal with the locomotion planning problem but only with the problem of designing the perception system, i.e. in which links, positions and orientations to physically attach sensors on the robot, considering the kinds of motion that the robot is expected to execute. Our sensor scheduling problem is nevertheless tightly related to the planning problems studied within the "active vision" and next-best-view-planning literature, which we turn to next.

Active vision and sensor scheduling algorithms

The field of active vision is concerned with the problem of controlling the viewpoint of a sensor such as to improve task performance. The problem dates back to the pioneering work of Aloimonos et al. (1988) and has been actively revisited since then Scott et al. (2003); Chen et al. (2011). This is essentially the same problem that we tackle in this paper: to decide which sensor to process at each point in time such as to improve mapping quality.

One approach to the problem is to incrementally compute and target a sensor at the Next-Best-View (NBV) according to some criteria related to task performance (e.g. reduce entropy in 3D reconstruction). For example, Brandao et al. (2013) proposes a simple NBV algorithm which greedily targets the gaze of a humanoid robot at points of maximum entropy along a robot trajectory. The NBV problem can also be formulated as sequential decision making under uncertainty, and framed within the reinforcement learning domain Sutton and Barto (1998). For example, de Figueiredo et al. (2018) formulates NBV as a Multi-Armed Bandit (MAB) problem Katehakis and Veinott Jr (1987), which allows to employ Bayesian Optimization (BO) to trade-off exploration and exploitation. In particular, they use confidence measures of 3D information and short-term egocentric memory to autonomously drive a robot's gaze direction during search tasks using BO.

Another body of literature frames NBV planning as a Partially Observable Markov Decision Process (POMDP) Ahmad and Yu (2013); Butko and Movellan (2010); Chong et al. (2008). Unlike MABs, these have the advantage of allowing non-myopic planning. One paradigm for solving POMDPs computes full policies offline through comprehensive evaluation and induction, before run-time Spaan (2012). Such offline methods are suitable for problems involving relatively small state, observation and action spaces, and achieve remarkable performance at the cost of low computation speed due to the curse-of-dimensionality and history Pineau et al. (2006). Online methods, on the other hand, start from the current belief state and simulate future rewards for finite planning horizons Ross et al. (2008); Browne et al. (2012) using Monte Carlo Tree Search (MCTS) techniques. Such methods can be run in real-time, as demonstrated for example by Figueiredo et al. (2017) in resource-constrained scenarios of multiple object tracking.

In this paper we also adopt a POMDP formulation to the problem of scheduling the next sensor(s) to process on a resource-constrained many-sensor robot. In particular, we use a POMDP formulation together with sampling techniques for fast planning, which is of utmost importance in the context of scheduling (i.e. the time spent on scheduling itself should not surpass sensor acquisition and processing time).

In the particular context of active 3D reconstruction Delmerico et al. (2018), existing NBV approaches belong to one of two main categories: frontier-based and information-driven planning. Frontier-based planners Yamauchi (1997); Dornhege and Kleiner (2013) guide the robot to boundaries between unknown and free space, which implicitly promotes exploration. Information-driven methods back-project probabilistic volumetric information on candidate views via ray casting, and select the views that maximize expected information gains Delmerico et al. (2018). Methods differ in the way they define information gain. For example, the authors of Kriegel et al. (2015) propose to use the average information theoretic entropy over all voxels traversed via ray casting. Instead of just considering the entropy, Isler et al. (2016) proposes a set of extensions to Kriegel et al. (2015)'s information gain definition, including the incorporation of visibility probability as well as the likelihood of seeing new parts of the object.

In this work, we adopt an information gain definition that similarly to the one of Palazzolo and Stachniss (2017) computes expected entropy losses, by accounting for known sensor characteristics. However, unlike common NBV planning approaches that plan for single-camera trajectories (e.g. Bircher et al. (2016)), our approach deals with the problem of maximizing mapping quality of multi-camera systems by scheduling the acquisition of cameras fixed in different locations of articulated multi-legged robots. Additionally, we consider an extra UCB term in the reward to prevent consistently sampling from problematic sensors that do not provide any information—this will prove crucial in the experimental evaluation.

Visual mapping and terrain modeling

In this paper we use probabilistic 3D occupancy grids for mapping and to guide sensor scheduling. Such maps represent the environment as a block of cells, each one having a binary state (either occupied, or free). They are popular in the robotics community since they simplify collision checking and path planning, access is fast and memory use can be made efficient through octrees Hornung et al. (2013).

Elevation maps Herbert et al. (1989) are a more compact 2.5D probabilistic representation that encode continuous heights on a 2D grid Michel et al. (2005), offering a convenient representation for legged locomotion Gutmann

et al. (2005). However, they are unsuitable for complex environments where the agent may have to navigate between objects at distinct heights (e.g. a ladder, a structure with several levels or floors). Multi-level surface maps Triebel et al. (2006) overcome this limitation by storing a list of heights for each cell grid. Despite being memory efficient, their main setback, resides in the impossibility of explicitly distinguishing between unknown and free space, which is essential for environment exploration and safe-navigation tasks. Recently, the idea of using continuous representations in mapping has also attracted great attention in the robotics community O'Callaghan et al. (2009). For example, O'Callaghan and Ramos (2012) uses Gaussian Processes (GPs) to encode interdependence between cells and thus correlations between structures in the environment. While continuous mapping based on GPs offers a convenient framework for exploration via Bayesian inference it still lacks computational efficiency, since it relies on BO techniques in high-dimensional spaces. Recent work of Jadidi et al. (2018) proposes the promising idea of considering fewer observations for close to real-time inference. However, GPs still require intensive sampling during collision checking for motion planners, which could be prohibitive for real-time applications.

In this paper we use the more mature voxel-based environment representation, in particular the OctoMap of Hornung et al. (2013), although in principle our efficient scheduling algorithm can be applied to GP-based maps as well.

Designing a many-sensor perception system

Consider a robot covered in depth sensors. At each of the robot's links, a set of locations where sensors can be attached is identified, and a sensor is attached at each location. Our approach in this section is to start from such an hypothetical robot and then decide which of the depth sensors to keep and which to scrap from the final design of the robot while at the same time optimizing the orientation of the sensors.

Definitions

More formally, let $\mathcal{C} = \{c_1, ..., c_K\}$ be a set of sensors indexed by $\mathcal{K} = \{1, ..., K\}$ where each c_k is a tuple $(l_k, x_k, \theta_k, u_k)$ which indicates which link l_k the sensor is attached to, the position $x_k \in \mathbb{R}^3$ and rotation $\theta_k \in [-90, 90] \times [-90, 90] \times] - 180, 180]$ degrees defined as the roll-pitch-yaw Euler angles of rotation of the sensor with respect to the link's local reference frame, and finally a flag $u_k \in \{0, 1\}$ indicating whether the sensor will actually be included in the final design of the robot or not. Note that although we use Euler angles, any other representation could be used. For simplicity we also assume all sensors to be equal (i.e. the same device) with equal shape, field-of-view, and near and far planes. We will later on describe how sensor locations are chosen.

We then define a set of tasks $\mathcal{Q} = \{Q_1, ..., Q_{|\mathcal{Q}|}\}$ that the robot is expected to execute, and which cover the distribution of possible robot motions. These could be instances of ladder-climbing tasks, crawling tasks, manipulation tasks, etc. Task number μ is a sequence of robot configurations $Q_\mu = [q_{\mu,1}; ...; q_{\mu,T\mu}] \in \mathbb{R}^{D \times T_\mu}$, where $q_{\mu,t} \in \mathbb{R}^D$ is configuration number t of task μ, D is the number of degrees of freedom of the robot and T_μ is the number of configurations in task μ. Additionally, for convenience we assume that any neighbor configurations in time (e.g. $q_{\mu,t}$ and $q_{\mu,t+1}$) belong to different stances, which in the context of this paper means that each new configuration either adds or removes a limb from contact. This will be useful when introducing task-specific coverage constraints later on.

Deciding the final design will involve several conflicting objectives, such as financial and assembly cost, data throughput, visual coverage and others. In this paper we focus on reducing the number of sensors used (which translates to assembly, financial and data costs) and increasing visual coverage (by choosing which of the candidate sensor locations to use and the orientation of those sensors). The number of sensors can be computed by a function

$$f_{\text{n}}(u_{1:K}) = \sum_{k=1}^{K} u_k,$$

where we explicitly write the dependency of the function f_{n} on the variables $u_1, ..., u_K$. Here and throughout the rest of paper we will use $u_{1:K}$ as a compact representation of all variables $u_1, ..., u_K$. Regarding visual coverage, in this paper we define it as the fraction of a set of points of interest that is visible from at least one sensor. We consider two types of coverage: spherical coverage $f_{\text{c\_sphere}}$ and task-specific coverage $f_{\text{c\_task}}$. In the case of spherical coverage, points lie on a large spherical surface centered on the base link of the robot. We sample these points uniformly on the surface, and so the points serve in this case as a way to measure the number of viewing directions that the perception system can cover. Coverage as a fraction of visible points is also the representation choice used by Mayol-Cuevas et al. (2009); Hörster and Lienhart (2006). Spherical (viewing direction) coverage is especially important in our problem since the robotic platform we will

use is targeted at versatile locomotion in challenging and unpredictable scenarios, where it can walk, crawl, or climb in any direction and facing up or down. Regarding task-specific coverage the idea is that, depending on the task, specific regions of space might be of crucial importance and thus should be given special consideration. Specifically, the points of interest for task-specific coverage are end-effector positions just before they are moved, i.e. end-effector positions which are planned to move at $q_{\mu,t+1}$ should be seen by at least one sensor at configuration $q_{\mu,t}$. The motivation is to allow for re-planning of contact placement just before contact, manipulation with visual servoing or virtual-reality interfaces, etc. More formally, we compute visual coverage as the expected value of point visibility, with respect to robot task and configurations, as:

$$\begin{aligned} f_{\text{c}} &= E[f_{\text{visible}}(y,\mu,q)] \\ &= \sum_{\mu=1}^{|\mathcal{Q}|} P(\mu) \sum_{t=1}^{T\mu} P(q_{\mu,t}|\mu) \frac{1}{|Y|} \sum_{y\in Y} f_{\text{visible}}(y,\mu,q_{\mu,t}), \end{aligned}$$

where Y is the set of points of interest, i.e. uniformly-sampled points on the sphere for spherical coverage $f_{\text{c\_sphere}}$, or the set of moved end-effector positions at $q_{\mu,t+1}$ for task-specific coverage $f_{\text{c\_task}}$. The function $f_{\text{visible}}(y,\mu,q_{\mu,t})$ is given by Algorithm 1 and returns 1 or 0 depending on whether point y is visible or not from configuration $q_{\mu,t}$ in task μ. The probabilities $P(\mu)$ and $P(q_{\mu,t}|\mu)$ are priors for the distribution of robot tasks and configurations on a specific task, respectively (i.e. how frequently you expect the robot to do a certain task and to be at a certain configuration during that task). In our experiments we use uniform distributions for both priors, $P(\mu) = 1/|\mathcal{Q}|$, $P(q_{\mu,t}|\mu) = 1/T_\mu$, in order not to favor or overfit the design to specific tasks.

Algorithm 1 POINT_VISIBILITY (pseudo-code for $f_{visible}(y,\mu,q)$)

input: sensor set $\mathcal{C}$; point of interest y; task number μ; robot configuration q
for SensorIteration $j \leftarrow 1, 2, ..., K$ **do**
  if $y \in$ FIELD_OF_VIEW(q, c_j) **and not** RAY-CAST_HITS(q, c_j, y) **then**
    return: 1 {point is visible by at least one sensor}
  end if
end for
return: 0 {point not visible}

Symmetry constraint

We enforce a symmetrical design, i.e. placement of sensors, to be consistent with the symmetrical shape of our robotic platform and to avoid overfitting specific tasks. As we will show later, this choice will also considerably speed up the optimization process because of a reduction in search space.

Our approach starts from asking a user to provide a symmetric robot configuration and a set of symmetry axes. Given this input, we visit each sensor in turn and identify its symmetrically placed sensors. For each set of symmetrically placed sensors, we will call one of them the "parent". Basically, the optimization will then be made only over variables of "parent" sensors, while the variables of other sensors will be implicitly given by the values of their parents.

Formally, we define a map $\pi : \mathcal{K} \to \{\mathcal{K} \cup \varnothing\}$, where $\pi(j) = k$ if sensor number k is a parent of sensor number j. A sensor will have no parents (i.e. $\pi(j) = \varnothing$) if it is already a parent of another sensor or if it has no symmetric counterparts. For the design to be symmetric, a sensor c_j should be included in the design when $c_{\pi(j)}$ is also included, which means we enforce the constraint $u_j = u_{\pi(j)}$ for all $j : \pi(j) \neq \varnothing$.

Rotations of symmetrically placed sensors should also be symmetric, for which purpose we define a rotation multiplier $f_{mult} : \mathcal{C} \to \{-1, 1\}^3$ which identifies whether the roll, pitch and yaw rotations of sensor c_j should be inverted or the same as its parents'. We thus enforce the constraint $\theta_j = \theta_{\pi(j)} \diamond f_{mult}(c_j)$ for all $j : \pi(j) \neq \varnothing$, where $\diamond$ represents component-wise multiplication.

We build the map π automatically from the symmetric robot configuration and symmetry axes provided by the user. Each sensor is visited in turn and its symmetrically placed sensors given the symmetry axes are identified. The identified sensors are marked as visited and their parents set appropriately. The procedure is run for all remaining unvisited sensors so that one parent will have multiple "children", but each "child" only one parent. The map f_{mult} is also built automatically and depends only on which plane the symmetry occurs.

Semi-automatic detection of possible sensor locations

Consider a bounding box of a robot's link aligned with the link's local reference frame. We will refer to each of this box's sides as a "face". Given a face of a robot's link, we generate possible sensor locations automatically in the following way. We distribute a grid of points uniformly on the face, separated by a distance equal to the maximum distance of the sensor shape's center to its edges, to avoid inter-sensor collision. Each point is then projected back to the 3D model of the robot link using raycasting along the

face's normal direction. Successful projections are added to $\mathcal{C}$, while raycasts that do not hit the original link are ignored.

To obtain the set of faces over which sensors are laid out we use a semi-automatic approach. We start by assuming all links could have sensors attached, and then we have a designer (mechanical engineer) flag those links or individual faces that are unusable—thus eliminating any sensors in those regions. This manual step takes advantage of the designer's know-how and can hardly be done automatically, since considerations such as space for cabling, possibility to open holes in parts, etc, need to be used to discard some of the sensor locations. Note that these considerations require knowledge of properties not included in usual robot models (i.e. materials, existing cables and holes inside parts, etc). For this paper, we also allowed the designer the option to manually add a sensor location to any link. This was done only on the feet links, since the feasible attachment locations are extremely constrained in that case - only one location per face.

Optimization problem and algorithm

We optimize the choice of sensors and their rotations through an optimization procedure. The problem is multi-objective and of the following form:

$$\underset{u_{1:K},\theta_{1:K}}{\text{minimize}} \ \left(\ f_{\text{n}}(u_{1:K}) \ , \ -f_{\text{c\_sphere}}(u_{1:K},\theta_{1:K}) \ \right) \tag{1a}$$

subject to

$$u_k \in \{0,1\} \quad \forall_{k\in 1,\dots,K} \tag{1b}$$

$$\underline{\theta}_k \leq \theta_k \leq \overline{\theta}_k \quad \forall_{k\in 1,\dots,K} \tag{1c}$$

$$u_k = u_{\pi(k)} \quad \forall_{k:\pi(k)\neq\varnothing} \tag{1d}$$

$$\theta_k = \theta_{\pi(k)} \diamond f_{mult}(c_k) \quad \forall_{k:\pi(k)\neq\varnothing} \tag{1e}$$

$$f_{\text{c\_task}}(u_{1:K},\theta_{1:K}) = 1, \tag{1f}$$

where $\underline{\theta}_k$, $\overline{\theta}_k$ are parameters representing the angle limits for sensor rotations. Note that we explicitly write the dependency of the coverage functions on the optimization variables for clarity.

Our approach to solving this problem is to compute the Pareto-front of the objective functions, and then allow a designer to pick one of the optimal designs using his or her intuitions. The designer's considerations could include different costs and benefits of each design, such as coverage gains per additional sensor, financial budget limits, implementation ease, or others. We opt for this approach for two reasons: 1) it reduces the room for gross designer errors since all solutions along the pareto-front are optimal; and 2) it alleviates the objective mis-specification problem, in particular of ignoring important designer considerations which are difficult to encode onto a single objective function. We will discuss limitations of this approach in the last section of the paper.

We solve problem (1) using NSGA-II (Non-dominated Sorting Genetic Algorithm II) Deb et al. (2002). NSGA-II is a genetic algorithm whose selection operator directs the population's individuals towards the Pareto-optimal front, obtains a good spread of solutions within the front, and has good empirical convergence properties compared to similar methods. As a genetic algorithm, it is also suitable to the non-convexity and non-differentiability of our problem.

Reformulation with penalties and symmetry

In practice, including nonlinear equality constraints such as our task-specific coverage constraints in genetic algorithms is not trivial since any perturbation in the optimization variables may bring them out of the feasible set. For this reason, we reformulate problem (1) into a simpler integer-and-bound-constrained problem. For this we define $\mathcal{K}' = \{k \in \mathcal{K} : \pi(k) = \varnothing\}$ as the set of all indices of sensors with no parents, i.e. those sufficient to define the whole design. Our new optimization variables will be $u'_{1:K'} = [u_k]_{k\in\mathcal{K}'}$ and $\theta'_{1:K'} = [\theta_k]_{k\in\mathcal{K}'}$, where $K' = |\mathcal{K}'|$. Finally, we move the task-specific coverage constraint to the objective function using a penalty method and solve the following optimization problem:

$$\underset{u'_{1:K'},\ \theta'_{1:K'}}{\text{minimize}} \ \left(\ f'_{\text{n}}(u'_{1:K'}), \tag{2a}$$

$$- f'_{\text{c\_sphere}}(u'_{1:K'},\theta'_{1:K'}) - \alpha f'_{\text{c\_task}}(u'_{1:K'},\theta'_{1:K'}) \ ) \tag{2b}$$

subject to

$$u'_k \in \{0,1\} \quad \forall_{k\in 1,\dots,K'} \tag{2c}$$

$$\underline{\theta}'_k \leq \theta'_k \leq \overline{\theta}'_k \quad \forall_{k\in 1,\dots,K'}, \tag{2d}$$

where the functions f'_{n}, $f'_{\text{c\_sphere}}$ and $f'_{\text{c\_task}}$ are new versions of the ones used in the original problem (1), but that will convert the optimization variables to the full variables $u_{1:K}$ and $\theta_{1:K}$ before running the rest of the operations. Additionally, $\alpha \gg 1$ is a penalty coefficient. It should be set to a large value (10^4 in our experiments) so that task-specific coverage works as a constraint in practice. This penalty function is a simple yet efficient way to include constraints in evolutionary algorithms Coello (2002). We refer the interested reader to Coello (2002) for a survey of alternatives.

In the remaining part of the paper, we will represent the solution of this problem that is finally implemented in the robot by $\mathcal{C}^*$, of size $K^* = |\mathcal{C}^*| = f'_{\text{n}}(u'^*_{1:K'})$.

Scheduling and mapping on a many-sensor perception system

Since the robot computational resources may not be sufficient to process all sensors at the same sampling step, processing must be multiplexed in time. The scheduling algorithm (see Algorithm 2), selects where to look next at each time (i.e. which sensors to process), to improve mapping performance under resource constraints.

Algorithm 2 Scheduling algorithm

input: grid map m; task prior $\mathcal{D}$; number of cameras K^*; maximum camera activation K_{max}; number of camera pixels to sample $F \leq N_p$
begin procedure
select $M < K^*$ cameras, considering $\mathcal{D}$ (eq. 17)
loop over selected cameras:
  uniformly sample $F \leq N_p$ camera pixels
  loop over pixels:
    compute expected information gain (eq. 10)
select top K_{max} expected reward cameras (eq. 14)
acquire point clouds from cameras and update m (eq. 6)
end procedure

Probabilistic volumetric occupancy mapping

Let us consider a map, defined as a 3D uniform voxel grid structure that encloses the workspace around the robot, represented by $m = \{m_i\}$, where each voxel $m_i \in \{0, 1\}$ is a binary random variable representing its occupancy. We use recursive Bayesian volumetric mapping Thrun et al. (2005) to sequentially estimate the posterior probability distribution over the map, given sensor measurements $z_{1:t}$ and sensor poses $p_{1:t}$ obtained through the robot forward kinematics model, from time 1 to t:

$$P(m|z_{1:t}, p_{1:t}) = \prod_i P(m_i|z_{1:t}, p_{1:t}) \tag{3}$$

assuming the occupancy of individual cells are independent. Filtering updates can be recursively computed in log-odds space Moravec and Elfes (1985) to ensure numerical stability and efficiency using the the following iterative probabilistic sensor fusion model

$$L(m_i|z_{1:t}, p_{1:t}) = L(m_i|z_{1:t-1}, p_{1:t-1}) + L(m_i|z_t, p_t) + L(m_i) \tag{4}$$

with

$$L(.) = \log\left[\frac{P(.)}{1 - P(.)}\right]$$

where $L(m_i|z_t, p_t)$ represents the inverse sensor model, $L(m_i|z_{1:t-1}, p_{1:t-1})$ represents the recursive term and $L(m_i)$ the prior. We assume that the map is initially unknown, i.e. $P(m_i) = 0.5$, eliminating the last term of equation (4).

Efficient probabilistic sensor model

At each time instant, each camera $c_k \in \mathcal{C}^*$ outputs a 3D point cloud $z_{t,k} = \{z^o_{t,k} \in \mathbb{R}^3, o = 1, ..., N_p\}$, defined in the reference frame of camera k with origin at the camera's optical center, where N_p is the number of camera pixels (assumed equal for all cameras). Let us consider the set of all ray traces from camera c_k, $\mathcal{G}_{t,k} = \{g^1_{t,k}, ..., g^{N_p}_{t,k}\}$, and the corresponding set of all traversed voxels, for a given ray trace g, $\mathcal{V}_{t,k,g} = \{v^1_{t,k,g}, ..., v^{N_v}_{t,k,g}\} \subset m$.

Our sensor noise model is based on the one proposed in Nguyen et al. (2012), which assumes that single point measurements $z^o_{t,k}$ are normally, independent and identically distributed (iid) according to

$$z^o_{t,k}|m, p_{t,k} \sim \mathcal{N}(z^{o*}_{t,k}; \Sigma^o_{t,k}) \tag{5}$$

with

$$\Sigma^o_{t,k} = \text{diag}(\sigma^l_{t,k}, \sigma^l_{t,k}, \sigma^a_{t,k})$$

where $z^{o*}_{t,k}$ denotes the true location of the measurement and $\sigma^l_{t,k}$ and $\sigma^a_{t,k}$ represent the lateral and axial noise standard deviations, respectively.

For the sake of computational complexity, we assume that noise is predominant in the axial direction ($\sigma^a >> \sigma^l$) which corresponds to the principal component of the covariance matrix. This assumption allows approximating the 3D covariance matrix by a 1D variance and rely on cheaper 1D cumulative distributions, for efficient probabilistic Gaussian fusion on 3D volumetric maps.

For each measurement $z^o_{t,k}$, we then update the corresponding closest grid cell m_i as occupied, and approximate the probability enclosed within the cell volume as follows

$$P(m_i|z^o_{t,k}, p_{t,k}) \approx F_z\left(z^o_{t,k} + \frac{\delta}{2}\right) - F_z\left(z^o_{t,k} - \frac{\delta}{2}\right) \tag{6}$$

where δ represents the grid resolution and $F_z(.)$ the cumulative normal distribution function of z, where the axial error standard deviation can be approximated by a simple quadratic model

$$\sigma^a_{t,k} \approx \lambda_a \|z^o_{t,k}\|^2 \tag{7}$$

with λ_a being a sensor specific scaling factor. All other cells belonging to the set of voxels traversed through ray casting (from the origin to end point z^o_i) are updated as being free with probability P_{free}. This is a reasonable approximation,

considering that map resolution and sensory noise have the same order, while at the same time allowing to significantly reduce the amount of computation.

Self-occlusion-aware expected entropy loss

Our scheduling algorithm tries to minimize reconstruction uncertainty by prioritizing the robot's limited resources to promising 3D regions, and does so by choosing actions which maximize rewards related to information gains. Let us assume that each camera has a deterministic, time-varying binary activation state

$$a_t \in \{a_t^k \in \mathbb{B}, k \in \{1, ..., K^*\}\} = \mathbb{B}^{K^*} \tag{8}$$

where $\mathbb{B} = \{0, 1\}$ with 0 and 1 meaning "off" and "on", respectively.

In our formulation, the reward signal $r_t^k(a_t^k) \in [0, 1]$ is dependent on camera activation according to

$$r_t^k(a_t^k) = \begin{cases} 0 & \text{if } a_t^k = 0 \\ \frac{1}{|\mathcal{G}_{t,k}|} \sum\limits_{\forall g \in \mathcal{G}_{t,k}} \frac{1}{|\mathcal{V}_{t,k,g}|} \sum\limits_{\forall v \in \mathcal{V}_{t,k,g}} I_t(v) & \text{if } a_t^k = 1 \end{cases} \tag{9}$$

where $|.|$ represents the set cardinality operator, and $I_t(v)$ represents the information gain for a given traversed voxel $v \in \mathcal{V}_{t,k}$, observed from camera c_k and at time t. We define this information as

$$I_t(v) = I_t^o(v) I_t^s(v) \tag{10}$$

where

$$I_t^o(v) = \mathcal{H}_{1:t}(v|z_{1:t}, p_{1:t}) - \mathcal{H}_t(v|z_{1:t-1}, p_{1:t-1}) \tag{11}$$

represents the entropy loss (i.e. information gain) after observing z_t, and where

$$I_t^s(v) = \begin{cases} 0 & \text{if } v \text{ is occluded by the robot} \\ 1 & \text{if } v \text{ is not occluded by the robot.} \end{cases} \tag{12}$$

is an indicator function which is activated only when voxel v is not occluded by the robot. We use raycasting as implemented in the ODE library for the self-occlusion computations ODE (2005).

The expected future entropy $E\left[\mathcal{H}_{t+1}(v|z_{1:t+1}, p_{1:t+1})\right]$ is obtained by considering the expected observation uncertainty given by the observation model and one recursion of the Bayesian filter, yielding

$$\begin{aligned} &E\left[\mathcal{H}_{t+1}(v|z_{1:t+1}, p_{1:t+1})\right] = \\ &= \mathcal{H}\left(L^{-1}\left(L(v|z_{1:t}, p_{1:t}) + E\left[L(v|z_{t+1}, p_{t+1}, a_{t+1})\right]\right)\right) \end{aligned} \tag{13}$$

Hence, unlike the reward formulation in Isler et al. (2016) which implicitly prioritizes resources to higher uncertainty regions, our proposed reward function encodes the expected decrease on entropy from observing a given voxel v from a given robot configuration p_{t+1}.

Information-driven scheduling under resource constraints

Like other information theoretical approaches that use uncertainty and task-related rewards to guide decision making, we frame our method within the reinforcement learning domain Sutton and Barto (1998). As such, the agent selects the perceptual actions (i.e. which sensors to use) such as to maximize expected cumulative rewards.

In the same spirit of Figueiredo et al. (2017), we formulate our online sensor scheduling task as an information maximization problem with resource allocation constraints as follows:

$$\begin{aligned} \underset{a_t}{\text{maximize}} \quad & R_t^T(a_t) = E\left[\sum_{\tau=1}^{T} \gamma^\tau \sum_{k=1}^{N} r_{t+\tau}^k(a_{t+\tau}^k) + \sqrt{\frac{2log(t)}{n_t^{a,k}}}\right] \\ \text{subject to} \quad & \sum_{k=1}^{N} a_t^k \leq K_{\text{max}} \quad \forall_{\tau \in \{1,...,T\}} \end{aligned} \tag{14}$$

where T is the planning horizon, $\gamma \in]0, 1]$ is a discount factor, K_{max} is the maximum number of active cameras per time-instant (i.e. camera activation capacity) and the square-root term is an upper-confidence bound (UCB) Auer et al. (2002) that handles the trade-off between exploration (minimizing uncertainty) and exploitation (maximizing rewards) Agrawal (1995). The vector $n_t^a = \left[n_t^{a,1}, ..., n_t^{a,K}\right]$ is a camera activation history accumulator where each element counts the number of times the respective camera has been activated so far, and the division is element-wise. We use this regularizing decaying term in order to deal with problematic scenarios such as cameras without any obstacles within the measurable range, or cameras which point at large surfaces of problematic properties (e.g. very bright surfaces in infrared sensors, no-texture in stereo). Using this UCB promotes exploration and avoids such cameras being consistently preferred.

Efficient sampling-based scheduling solution Efficient sampling algorithms are of utmost importance for real-time complex event processing and decision making problems. When dealing with large data streams, one would like to analyze only a partial amount of the data to cope with processing capacity, throughput and timing constraints.

In order to reduce the computational effort during planning (i.e. sensor scheduling computation), at each planning step t we consider only a subset of the total number of cameras and rays cast from each camera. The subset of raycasts is of size $F \leq N_p$ and is sampled without replacement from a uniform distribution in $\mathcal{O}(F)$ time. The subset of cameras is sampled from a discrete weighted distribution $\mathcal{W}_c = \{w_c^1, ..., w_c^{K^*}\}$. With scalability in mind, we rely on the reservoir sampling algorithm Exp-J proposed in Efraimidis and Spirakis (2006) for drawing without repetitions a weighted random sample of size M, in our case representing a camera sampling constraint during planning, from a population of size K^*, in $\mathcal{O}(M\log(K^*/M))$ time. This weighted-distribution formulation is general enough that task biases can be introduced into planning, which can be useful for example if prior knowledge on terrain geometry such as ceiling height is known - to avoid spending computational resources on cameras which point at regions of less importance for locomotion. These could be simple uniform or more relevant (learned or hand-designed) priors.

To account for such task-related sampling biases we adopt the unit sphere of orientations proposed in de Figueiredo et al. (2018), which is a convenient and flexible structure for encoding task-dependent orientation priors in an egocentric reference frame, and promote more frequent sampling to specific egocentrically encoded directions. The unit sphere of orientations is centered in the robot base link, and comprises a set of 3-dimensional unitary norm points, representing viewpoint directions,

$$\mathcal{D} = \{\mathbf{d}^k \in \mathbb{R}^3, k = 1, ..., N_d : \|\mathbf{d}^k\| = 1\} \quad (15)$$

which are i.i.d. and sampled from a Gaussian Mixture Model (GMM),

$$\mathbf{d}^k = \frac{\mathbf{v}^k}{\|\mathbf{v}^k\|} \text{ where } \mathbf{v}^k \sim P(\mathbf{v}) = \sum_{n_g=1}^{N_g} \phi^{n_g} \mathcal{N}(\boldsymbol{\mu}^{n_g}, \boldsymbol{\Sigma}^{n_g}) \quad (16)$$

where N_g represents the number of Gaussian components and ϕ^{n_g} the weight of the n_g-th Gaussian. The statistics of the GMM must be chosen according to the walking pattern and prior knowledge on terrain characteristics. If, for instance, the task is to walk ahead on a rough terrain, one could prioritize awareness to regions below and in front of the robot, which can be achieved by sampling from a two-component Gaussian Mixture, one with a larger mean in the bottom direction and other in the frontal direction (see Fig. 17). While if the task is, for instance, to walk through a narrow passage, one could prioritize attention to the sides, which can be achieved by sampling from a single component Gaussian, with larger variance in the lateral direction. These priors could also be left at a uniform distribution when such information is not available. The crucial point here is that the method allows to encode such priors. The i-th camera weight is proportional to the following sum:

$$w_c^i \propto \sum_{k=1}^{N_d} \mathbf{d}^k \cdot \mathbf{c}^i \quad (17)$$

where $\mathbf{c}^i \in \mathbb{R}$ represents the camera optical axis direction.

Results

Designing a many-sensor perception system

Robotic platform and target tasks We applied the design optimization methodology described in this paper to the robot WAREC-1 Hashimoto et al. (2017), which is shown in Figure 1. WAREC-1 is a versatile legged robot targeted at disaster response scenarios with high-power tasks in mind. It weighs 150kg, and its custom-designed actuators allow for climbing ladders and carrying heavy objects. The robot can move in different modalities, such as crawling, "belly-crawling" (i.e. crawling with alternating body and end-effector contact phases), quadrupedal walking, bipedal walking, wheeled locomotion (using modular attachable wheels) and ladder-climbing. It is also symmetric and can thus execute such modalities in different directions in principle (i.e. inverting "feet" and "hands" or "front" and "back").

We chose a set $\mathcal{Q}$ of robot tasks, or motion exemplars, according to these capabilities. To do so, we used robot motions previously executed in field trials and their simulations: wheeled motion, a manipulation motion, a crawling motion, and a ladder-climbing motion. We used 1, 2, 2 and 5 robot configurations respectively, which was the number of spline waypoints required to describe end-effector trajectories. As mentioned previously, each waypoint either adds or removes a limb from contact with respect to its following and previous waypoints. Each task was given the same total probability of 0.25. We also use a uniform distribution of configuration probabilities per task which

means that, for example, each ladder configuration is given probability 0.05. We show the configurations in Figure 2. While the number of postures and tasks is too low to model an accurate probability distribution, it is an approximation which represents the most common configurations of the robot when using our existing planners and controllers.

The result of the semi-automatic generation of the sensor set is shown in Figure 3. Sensors on the belly and some of the arm links were excluded because of attachment difficulty or to avoid reductions in movable joint range. These exclusions were made by qualitative judgements of our mechanical engineer. In particular, the belly of the robot (i.e. ground-looking face of the trunk) can make contact with the floor and thus the sensors could easily be damaged if placed here. Additionally, the "shoulder" links and some of the faces of the "knee" links were described as problematic for assembly by the engineer, due to drilling and physical space constraints. The final set of sensors used within the optimization is of size K=47.

Choice of sensor We opted for using the sensor CamBoard pico flexx from pmdtechnologies in our system design. The choice was influenced by several factors:

1. Possibility of using multiple (10s of) sensors simultaneously and on overlapping views, both in terms of firmware limitations and data throughput.
2. Availability of ROS drivers
3. Size and cost

Importantly, pico flexx was the sensor we found which allowed for the highest number of simultaneously connected sensors, because of image size, firmware and sensor design. Other sensors we considered were Kinect, Kinect 2, Xtion 2, Stereolabs' ZED and ZED mini, Code Labs' Duo and others. The sensor was also small (68x17x7.25 mm) and affordable compared to other available sensors in the market, and had a similar field-of-view (although slightly narrower than e.g. Xtion2 62x45 vs 74x52 degrees). The camera produces organized point clouds of size 224x171 at 5 to 45Hz using infrared sensing. The parameters we used in the optimization process and evaluation simulations were those in the sensor's datasheet: 62 x 45 degrees field-of-view, and 0.1 to 4.0 meter range.

Implementation We used OpenRAVE Diankov (2010) for computing forward kinematics and the Open Dynamics Library ODE (2005) for raycasting. To check whether a point falls inside the field-of-view of a camera we used simple geometrical checks: of whether it falls inside the rectangular pyramid given by the datasheet FOV angles and whether its distance is within the measurable range. We generated points on a sphere of 10 meter radius for visual spherical coverage computations by uniformly sampling θ and $cos(\phi)$, where θ and ϕ are spherical coordinates. After generating the points we kept them fixed throughout the whole optimization procedure. Figure 4 shows the sampled sphere.

For optimization we used DEAP Fortin et al. (2012), a Python library for evolutionary algorithms, along with its implementation of NSGA-II. We set all parameters to their default values, and paralelized the optimization over 12 threads (6 cores) of a 3.3GHz desktop computer using the distributed processing library SCOOP Hold-Geoffroy et al. (2014). We set the sensor rotation angle limits to $[-65, 65]$ degrees for pitch and $[-10, 10]$ degrees for yaw, in order to avoid reducing the movable range of the robot's joints taking into consideration the space requirements for the sensor and its holder. We fixed roll angles to zero for simplicity, to reduce the number of variables, and also to keep low the volume of new robot-model convex hulls (i.e. link plus sensors).

Baselines In order to allow the comparison of our system's performance with state-of-the-art systems, we use a set of "design approaches" of recent robotic platforms as baselines. By "design approach" we mean a combination of sensor placement and field-of-view. We selected and adapted existing designs from the literature and internet, with a focus on high potential spherical coverage. In particular, in this paper we consider the following approaches, all shown in Figure 5:

- "robosimian"-inspired approach, with multiple depth cameras around the body. Using a crawling posture as reference we use front/back-looking cameras (2), 45 degree downward looking cameras to the sides and front/back (4), as well as downward looking cameras (2);
- "robosimian-dbl", same as previous but with upward-looking cameras symmetric to the downward-looking ones.
- "spotmini"-inspired approach, with a forward-looking depth camera on the body and Velodyne-like 180x45 degree coverage on top of the body;
- "spotmini-new", inspired by a newer version of Spot Mini, with one depth camera at each side of the body;
- "e2dr"-inspired approach, a single head system with wide-coverage. We use a forward-looking depth

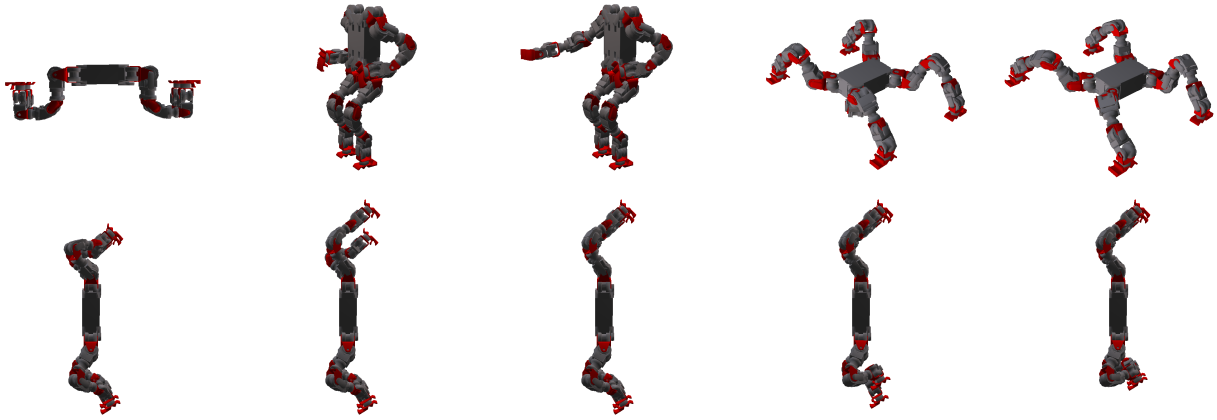


Figure 2. Robot configurations used for design optimization of the sensor system. From left to right, top to bottom: wheeled task (1 configuration), manipulation task (2), crawling task (2) and ladder-climbing task (5). In our experiments, each task as equal probability.

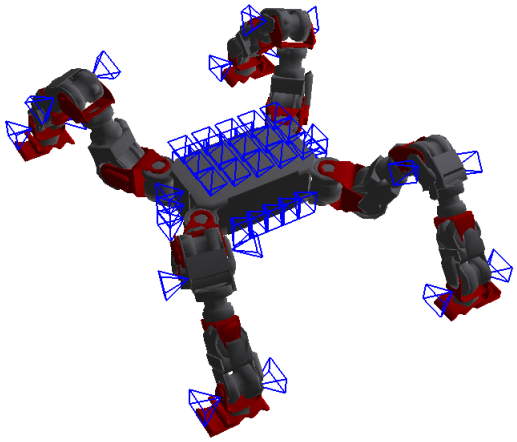


Figure 3. The set of 47 sensors used for optimization. Both sensor orientation and usage are optimized (i.e. roll-pitch-yaw angles and binary variables indicating whether each sensor is used in the final design or not).

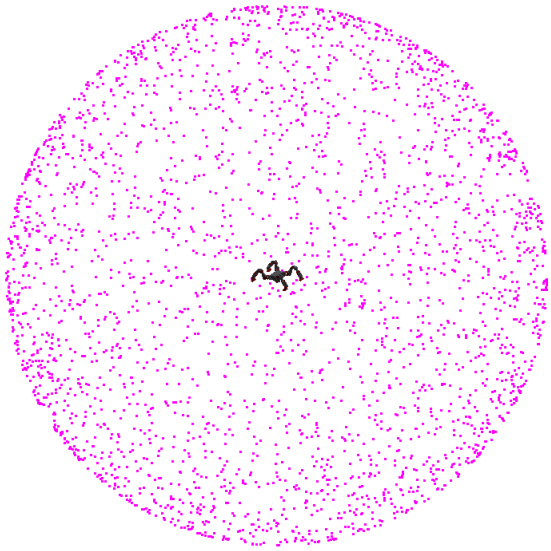


Figure 4. Robot and coverage sphere. In our sensor placement algorithm we optimize the number of viewing directions covered by the sensors, represented by the fraction of points visible on the surface of a large sphere.

camera and 180x180 degree coverage on each side. We also add a backward-looking depth camera for extra coverage (not present in e2dr);

- "e2dr-dbl", same as the previous but with a duplicate head at the bottom of the body as well for symmetry.

Note that for a fair comparison we replace laser rangefinders by a set of depth cameras covering approximately the same field-of-view. This is because we are interested in visual coverage at any given time, in which case laser rangefinders would only return points on a plane and not the whole field-of-view. Therefore, our evaluation will show higher coverage values than the original designs, and we stress that it is used for comparison not with the original robots but with their underlying sensor field-of-view and placement approach.

Optimization results We ran the optimization for 1500 generations, after which there were no visible changes in fitness values. We used populations of 300 individuals. The initial population was sampled randomly from a uniform distribution within variable bounds. Figure 6 shows the coverage and number of sensors of the initial and final populations. Figure 6 also compares the estimated Pareto front with the baseline design approaches. Interestingly, most of the baseline design approaches fall on the Pareto front, which means that state-of-the-art approaches are already optimal for WAREC-1, or at least close to optimal, in terms of placement and field-of-view. In other words, our approach was not able to obtain higher coverage for the same amount of sensors used, hence suggesting state-of-the-art approaches

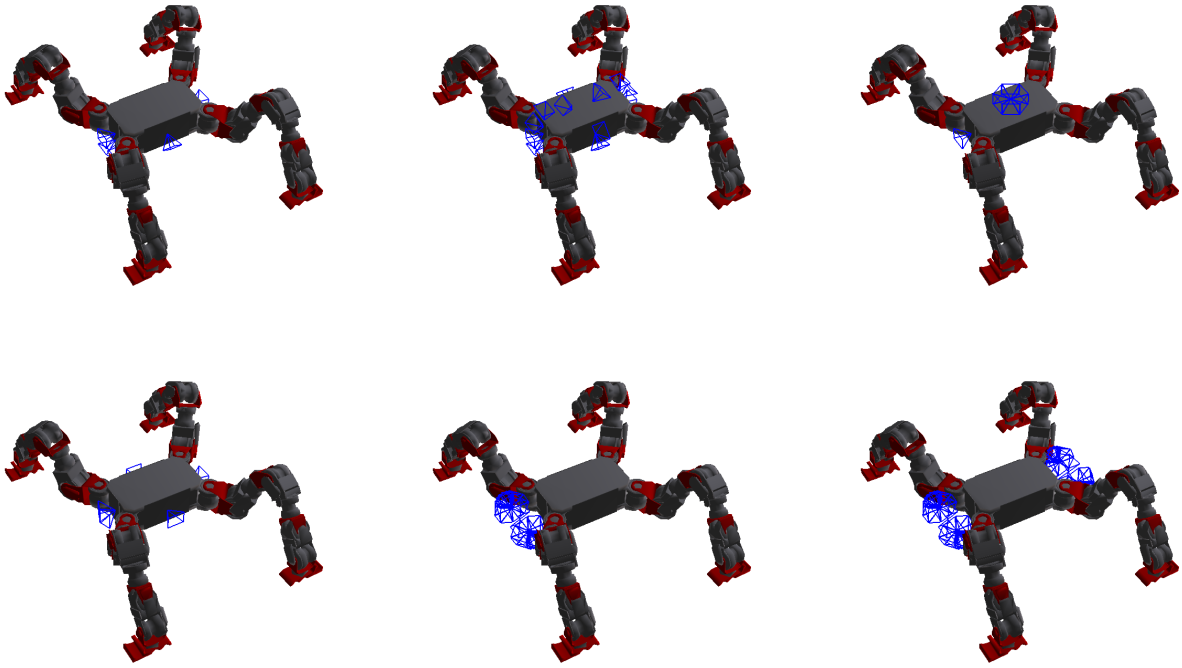


Figure 5. Design approaches used for comparison. From left to right, top to bottom: robosimian, robosimian-dbl, spotmini, spotmini-new, e2dr, e2dr-dbl.

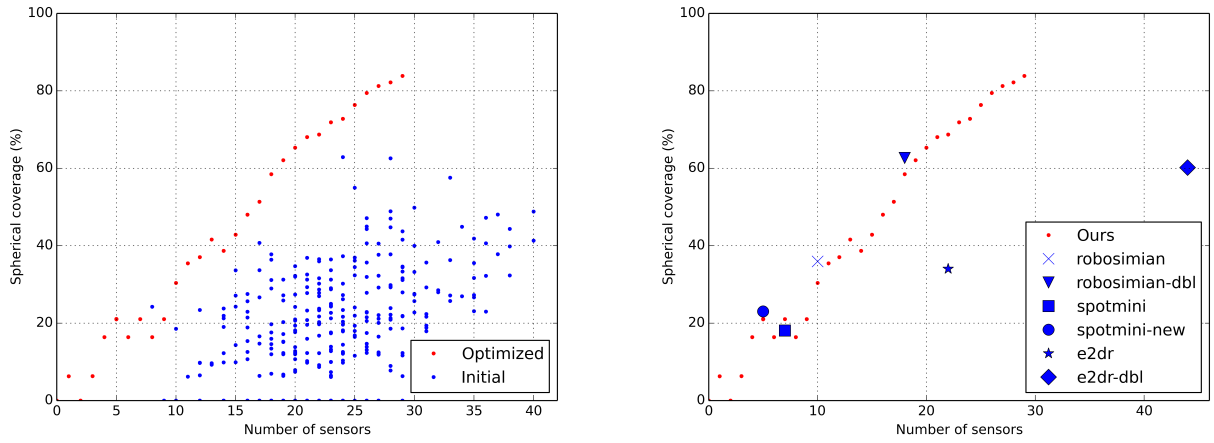


Figure 6. Left: initial and final population. Right: comparison with other design approaches.

place sensors in locations with the least number of self-occlusions. The exceptions were e2dr and e2dr-dbl, which performed poorly in terms of number of sensors in our robot since they rely heavily on laser rangefinders and have much occlusion with the shoulders and arms (which are larger than the actual e2dr robot's). While most baselines scored between 20 and 30% coverage, robosimian-dbl obtained a very high 60% coverage with only 18 sensors.

Figure 6 also shows that while baseline designs reached a maximum of 60% coverage, our Pareto-based approach also managed to find solutions of higher coverage - up to 80% even if no sensors were allowed on the robot's belly. Our solutions, as shown in Figure 7, manage to compensate for the lack of sensors on the belly by using downward and upward looking sensors on the feet, as well as downward-looking sensors on the knees and elbows.

Finally, we evaluate the advantage of using the symmetry reformulation (2). As shown in Table 1, using symmetry reduces the number of variables to about a third (47 cameras to 16 parent cameras). The reduction in search space is also associated with a higher maximum coverage and maximum coverage-per-sensor, of around 10%. Note that we obtained the results in the table using the same optimization parameters and running NSGA-II for

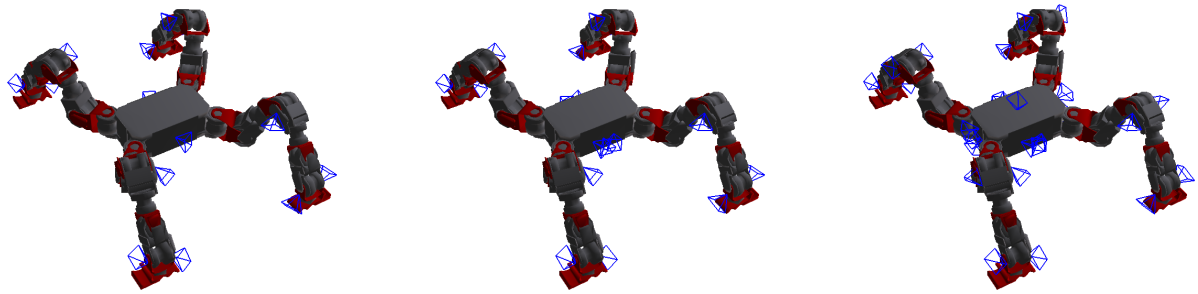


Figure 7. Examples of solutions in the final population. From left to right: number of sensors 14, 18, 29.

the same number of generations (1500). Also note that the symmetry constraint does not reduce the number of required raycasts since we do not require motions to be symmetric and as such we still compute coverage on all sensors. There is actually additional computation involved in the symmetry-constrained algorithm, of inferring children sensors' parameters from their parents'. This is reflected in higher computational time (4173 vs 3369 seconds for 1500 generations). We believe this gap could be made smaller with a smarter implementation, which in our case recomputed the full states from reduced states several times per fitness evaluation. We should note that the range of the number of sensors and of coverage is lower in the no-symmetry case because we use a fixed number of generations. In principle, the ranges obtained by NSGA-II should increase with additional time, however at the cost of longer computation times due to the 3x larger search space. In other words, the results in Table 1 show that using symmetry constraints leads to faster convergence.

Picking and evaluating the final design We had a mechanical engineer go over the optimal solutions along the Pareto front and pick one according to ease of attachment on the real robot and financial budget. We picked the solution with f_n=20 sensors, shown in Figure 8.

Table 2 shows spherical and task-specific (i.e. contact point) coverage on the optimized tasks, for both our final design and the baseline design approaches. Most noticeably, other designs often obtain 0% task-specific coverage, for an average of 11 to 33% over all tasks. The exception is e2dr-dbl (f_n=44), which obtains 66% task-specific coverage on average. Our final design obtains 100% coverage of contacts in all tasks. Regarding spherical coverage, our design has the highest expected value of 65%, and robosimian-dbl obtains 63%. Note that the latter uses sensors on the belly, which is difficult to implement in our physical robot in practice, since the robot contacts the ground with the base

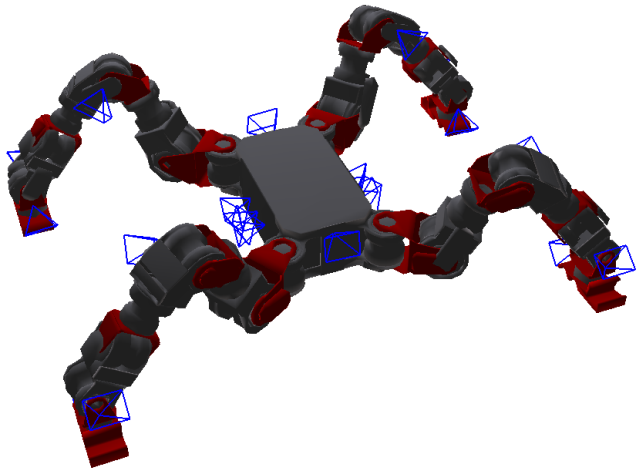


Figure 8. Final design of the robot's depth sensors.

link itself in some locomotion styles (e.g. belly-crawl). That is the reason why we excluded sensors in this area during optimization. Nevertheless, our design still manages to obtain slightly-higher coverage than robosimian-dbl by relying on leg/arm-attached sensors. The other designs performed poorly, scoring between 18 and 36% spherical coverage for $f_n \leq$22 and 60% for f_n=44.

Table 3 evaluates and compares our final design on a set of new tasks. Namely, we use a longer manipulation sequence (34 configurations) and crawling sequence (200), a new quadruped-walking motion on a plane (35), on stairs (118), on a tall step (41), and a biped-to-quadruped transition motion (14). The evaluation shows that our design's coverage in these tasks is very similar to that obtained on the optimized tasks (68 vs 65%). This shows that the optimized design is generalizable to new tasks and does not seem to have overfitted the original tasks. The next-best performance is again by robosimian-dbl. Interestingly, the coverage of some designs (e2dr and e2dr-dbl) is highly dependent on the task, varying within as much as 12 to 67% and 23 to 89%

Table 1. Influence of symmetry reformulation

Condition	NumVars	Solved	Time (s)	NumSensors	Coverage	Coverage/NumSensors
Ours	16*3	Yes	4173	0 to 29	0.00 to 82.87	0.00 to 6.25
Ours no symmetry	47*3	Yes	3369	0 to 19	0.00 to 73.25	0.00 to 5.53

Table 2. Coverage (%) comparison with other design approaches, on optimized tasks

Spherical coverage

Task \ Design	ours (f_n=20)	robosimian (f_n=10)	robosimian-dbl (f_n=18)	spotmini (f_n=7)	spotmini-new (f_n=5)	e2dr (f_n=22)	e2dr-dbl (f_n=44)
Wheeled	**67.57**	31.83	59.70	19.43	19.93	10.47	12.33
Manipulation	63.42	35.20	58.52	17.30	19.32	11.88	**79.83**
Crawl	66.90	40.37	68.23	17.68	28.80	66.83	**89.07**
Ladder	63.36	36.37	**64.23**	17.88	23.97	46.86	59.45
All	**65.31**	35.94	62.67	18.07	23.00	34.01	60.17

Task-specific (contact) coverage

Task \ Design	ours (f_n=20)	robosimian (f_n=10)	robosimian-dbl (f_n=18)	spotmini (f_n=7)	spotmini-new (f_n=5)	e2dr (f_n=22)	e2dr-dbl (f_n=44)
Wheeled	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Manipulation	**100.00**	**100.00**	**100.00**	0.00	0.00	0.00	**100.00**
Crawl	**100.00**	0.00	0.00	75.00	25.00	0.00	75.00
Ladder	**100.00**	0.00	0.00	0.00	0.00	50.00	50.00
All	**100.00**	11.11	11.11	33.33	11.11	22.22	66.67

respectively. This indicates that some of the baselines we tested could actually be preferred for robots of specific and fixed locomotion styles.

Finally, we evaluate the influence on performance of the choice of tasks used within the optimization. To do this, we computed the coverage obtained when only a single task is optimized (Table 4), and when all but one task are optimized (Table 5). As before, all results shown in the tables are for a 20-sensor design. The tables show that spherical coverage is similar although slightly higher (2-3 percentage points) when fewer tasks are considered. Except for the ladder task, spherical coverage is basically the same whether a single or all-but-one task are optimized. As seen previously in Figure 6, spherical coverage is tightly related to the number of sensors. Although the contact coverage constraint is demanding, it it not demanding enough to considerably influence the spherical-coverage-per-sensor. The ladder task is an exception and is more demanding than other tasks, as can be seen by the following points: 1) optimizing for the ladder task alone leads to high satisfaction of other tasks as well; 2) solving for all but the ladder task leads to higher spherical coverage (8 percentage points). As a comparison, this amount of spherical coverage increase corresponds approximately to an increase of 3 sensors in the design optimizing all tasks (Figure 6). Table 5 basically shows that contact coverage constraints are demanding on the system—as not considering a task will typically lead to the constraint not being satisfied.

Real robot implementation Figure 9 shows the final robot and a close-up of the installed sensors. We managed to keep all cabling and processing inside the robot, in the following way. First, we placed one USB hub inside of each joint and ran cables inside the links and motors connecting each consecutive pair of USB hubs. Then, we individually designed in SolidWorks and 3D-printed one holder for each sensor with the appropriate rotation angles with respect to the link it was attached to. We opened holes in the trunk link to pass sensor cables through, and installed 2 PCs* inside the trunk link collecting half the sensor data throughput each. With the double-PC setup it is possible to simultaneously acquire data from all sensors, although at the cost of virtually using all of the CPUs' capacity (roughly 60% for acquisition, 40% for cloud filtering). In the next section we will evaluate how scheduling can reduce this cost.

Figure 10 shows the robot walking on quadruped and crawling styles while simultaneously acquiring data from all sensors. As expected from the simulation-based evaluation,

*Intel NUC Kit NUC5i7RYH, which have an i7-5557U processor (up to 3.40GHz), 16GB memory and weight 1.1kg.

Table 3. Spherical coverage (%) comparison with other design approaches, on new tasks

Task \ Design	ours (f_n=20)	robosimian (f_n=10)	robosimian-dbl (f_n=18)	spotmini (f_n=7)	spotmini-new (f_n=5)	e2dr (f_n=22)	e2dr-dbl (f_n=44)
Manipulation long	63.46	34.74	59.29	18.35	18.89	11.85	**78.84**
Crawl long	66.73	40.30	68.16	18.03	28.83	66.94	**89.13**
Quadruped plane	**71.62**	33.98	61.85	20.09	23.01	18.45	23.31
Quadruped stairs	**68.82**	34.33	62.19	20.06	23.04	23.20	32.38
Quadruped step	**70.70**	32.15	60.02	19.89	21.48	17.76	20.82
Biped2quadruped	**69.12**	38.44	66.31	20.50	26.77	29.54	61.27
All	**68.41**	35.66	62.97	19.49	23.67	27.96	50.96

Table 4. Algorithm sensitivity: Coverage (%) on 20-sensor designs that optimize a single task

Spherical coverage

Evaluated task \ Design	Solving for all tasks	Solving for wheeled	Solving for manipulation	Solving for crawl	Solving for ladder
Wheeled	67.57	**79.53**	63.60	58.63	63.60
Manipulation	63.42	69.00	70.40	**72.38**	69.75
Crawl	66.90	63.07	56.52	**82.83**	59.63
Ladder	63.36	61.41	61.96	64.90	**71.43**
All	65.31	68.25	63.12	**69.69**	66.10

Task-specific (contact) coverage

Evaluated task \ Design	Solving for all tasks	Solving for wheeled	Solving for manipulation	Solving for crawl	Solving for ladder
Wheeled	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Manipulation	**100.00**	0.00	**100.00**	0.00	0.00
Crawl	**100.00**	25.00	50.00	**100.00**	25.00
Ladder	**100.00**	0.00	0.00	0.00	**100.00**
All	**100.00**	11.11	33.33	44.44	55.56

Table 5. Algorithm sensitivity: Coverage (%) on 20-sensor designs that optimize all but one task

Spherical coverage

Evaluated task \ Design	Solving for all tasks	Solving for all but wheeled	Solving for all but manipulation	Solving for all but crawl	Solving for all but ladder
Wheeled	67.57	59.83	69.60	76.30	**76.53**
Manipulation	63.42	67.52	64.12	68.92	**75.52**
Crawl	66.90	**80.77**	75.65	62.65	78.82
Ladder	63.36	64.42	69.97	67.38	**75.33**
All	65.31	68.13	69.83	68.81	**76.55**

Task-specific (contact) coverage

Evaluated task \ Design	Solving for all tasks	Solving for all but wheeled	Solving for all but manipulation	Solving for all but crawl	Solving for all but ladder
Wheeled	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Manipulation	**100.00**	**100.00**	0.00	**100.00**	**100.00**
Crawl	**100.00**	**100.00**	**100.00**	25.00	**100.00**
Ladder	**100.00**	**100.00**	**100.00**	**100.00**	0.00
All	**100.00**	**100.00**	88.89	66.67	55.56

the real system also has wide coverage. The figure shows that both walls and obstacles in all directions, as well as the ceiling can be acquired simultaneously for wide coverage. One concern we had with these results was that when all feet

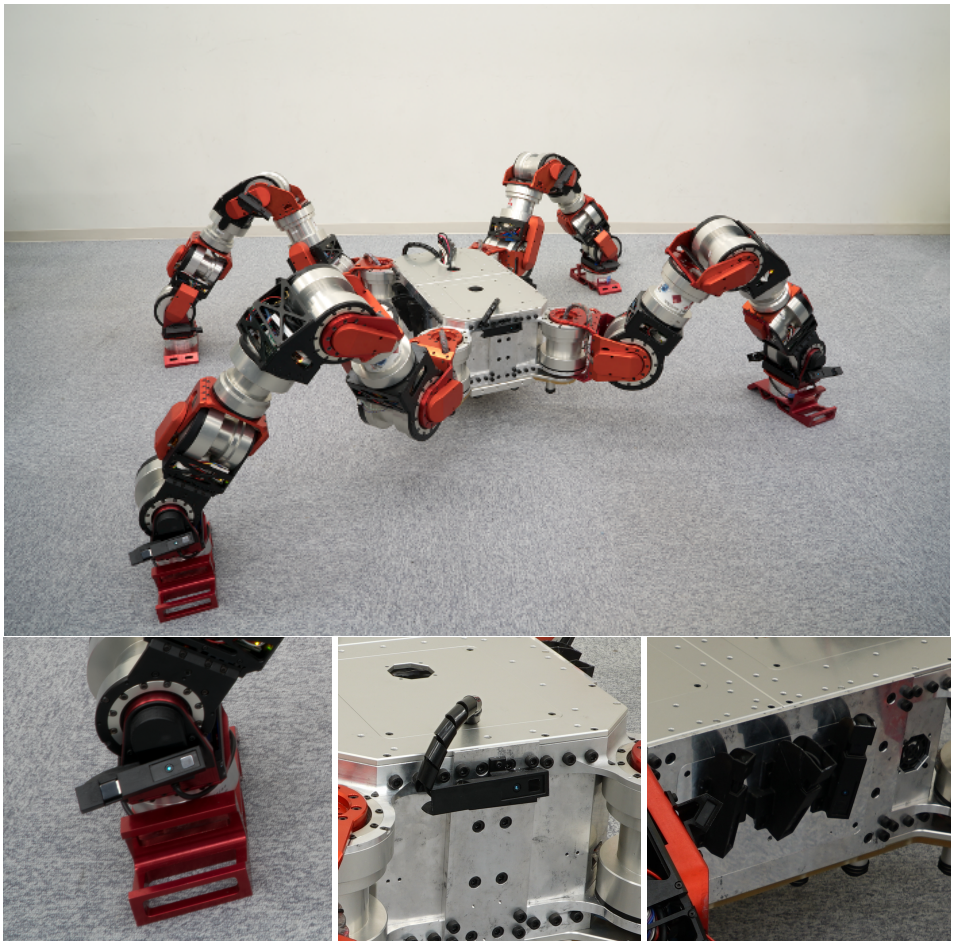


Figure 9. Final robot and a close-up of the camera holders.

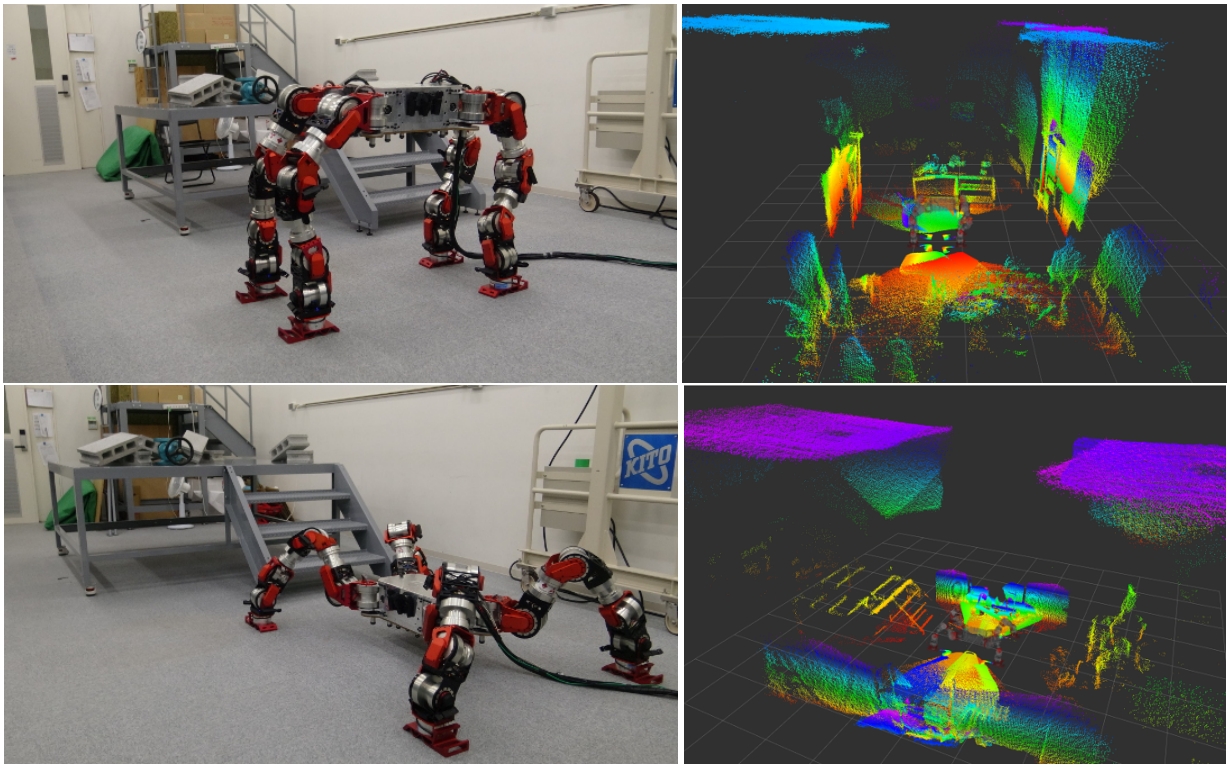


Figure 10. Point clouds of the full sensor system, when all sensors are acquired simultaneously (no scheduling). Top: quadruped, bottom: crawling locomotion.

are in contact, there are two blind-spots with no points on the floor plane, along the forward-backward axis. We believe the issue can be alleviated through sideways locomotion plans (which is possible in our platform) or by integrating information on a map (next Section). However, in Figure 11 we show that when the robot lifts its legs, the coverage of the floor plane is increased, as well as the area for the next contact point. This is obtained by design, since our optimization problem (2) included a task-specific constraint (i.e. contact point visibility from the previous stance).

We estimated point density at planned contact points (i.e. number of cloud points on the area of contact) to evaluate how well the task-specific constraint is respected on real scenarios. We measured point density both on the previous quadruped walking experiment (which was not part of the tasks in the optimization problem), and on a crawling experiment on top of rubble. Figure 12 shows the crawling experiment and interface for estimating point density. Using a foot-shaped box to count points at the next contact position, we measured an average of just above 1000 points, which is approximately 5 points per cm^2 in WAREC-1. Such density is higher than what is assumed at the contact planner level Brandao et al. (2016) (0.5 points per cm^2) and is enough to accurately estimate normal vectors from nearest-neighbor queries (which requires 20 neighbors or around 4 cm^2 per point in our current system). We will return to a further evaluation of the real system later, after showing results of the scheduling system in simulation on the next section.

Scheduling and mapping on a many-sensor perception system

In order to analyze the performance of the proposed information-driven sensor scheduler for active mapping, we first conducted a set of experiments in the Gazebo simulator Koenig and Howard (2004).

We considered a simulation setup (See Figure 13), that incorporates multiple challenges that a smart dynamic scheduling camera system has to face, including fast self-motion and no-visibility viewpoints. In this scenario, the robot is standing in front of a ladder, while moving its right front leg up and down such as to be able to see the step surfaces from above. Furthermore, there is no ceiling or walls and therefore some of the cameras do not return any points. An intelligent active mapping algorithm should exploit this behavior such as to maximize task-related rewards, in this case information gathering, by focusing on new rewarding viewing directions as soon as they become available and avoiding sensors with low visibility.

In all our experiments, the axial noise standard deviation scaling factor was set to $\lambda_a = 0.005$ and the occupancy probability threshold was set to $P_{occ} = 0.7$. In each experiment we let the observer collect $T = 100$ observations (i.e. planning and sensing iterations). Each experiment was repeated 10 times to average out variability in different simulations, due to the randomized nature of our algorithm, and non-repeatability influenced by multiple simulation factors including separate threads for Gazebo's physics and sensor generation, as well as stochastic latencies involved in inter-process communication.

Reconstruction Performance Metrics We focused our performance evaluation in the following metrics, which are common in the NBV planning literature:

- the temporal information gain (or temporal entropy reduction):

$$IG_t = \sum_{m_i \in m} \mathcal{H}_t(m_i) \tag{18}$$

which is a quality performance measure of the knowledge regarding the surrounding environment, gathered in the probabilistic volumetric map m, up to time t.

When normalized by the number of planning (i.e. sensor scheduling) steps it represents the temporal average global information gain per reconstruction step (i.e. sensor acquisition and insertion into the occupancy grid):

$$IG/S = \frac{1}{T} \sum_{t=1}^{T} IG_t \tag{19}$$

- amount of occupied cells (surface coverage)

$$SC_t = \sum_{m_i \in m} \mathbf{1}\left(P_t(m_i)\right) \tag{20}$$

where

$$\mathbf{1}\left(P_t(m_i)\right) = \begin{cases} 0 & \text{if } P_t(m_i) < P_{\text{occ}} \\ 1 & \text{if } P_t(m_i) \geq P_{\text{occ}} \end{cases}$$

which is a measure of task-completeness and quantifies the surface covered during reconstruction, where P_{occ} represents the user specified probability threshold of the volume being occupied.

When normalized by the number of reconstruction steps it represents the average surface coverage per

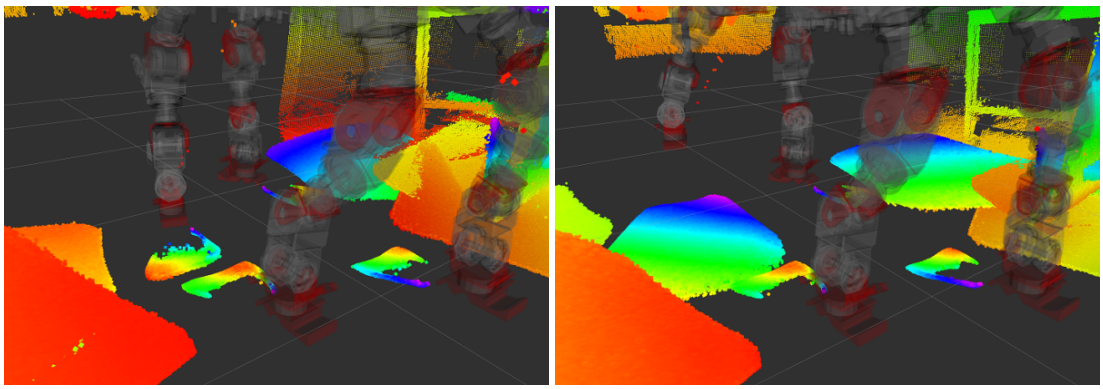


Figure 11. Simultaneously acquired point clouds, during quadruped locomotion. Left: all feet are in contact. Right: the robot lifts the forward-left foot before contacting the floor again. Coverage of the contact area increases while the leg is lifted.

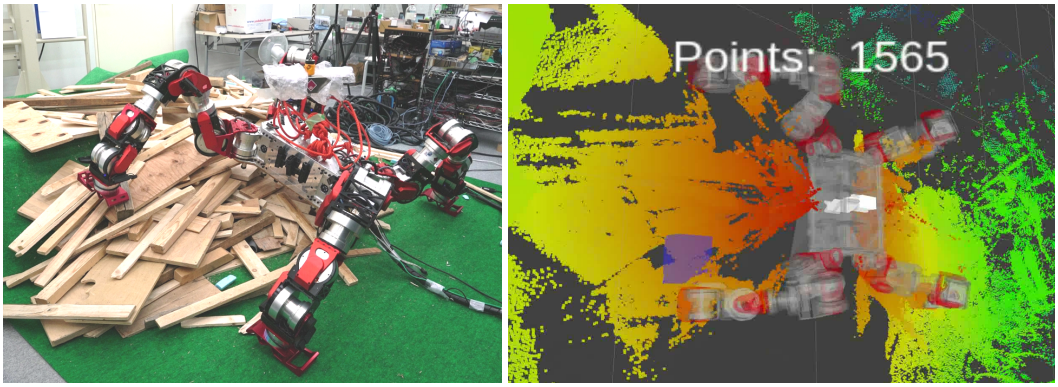


Figure 12. Point cloud density at the next contact point (task-specific coverage).

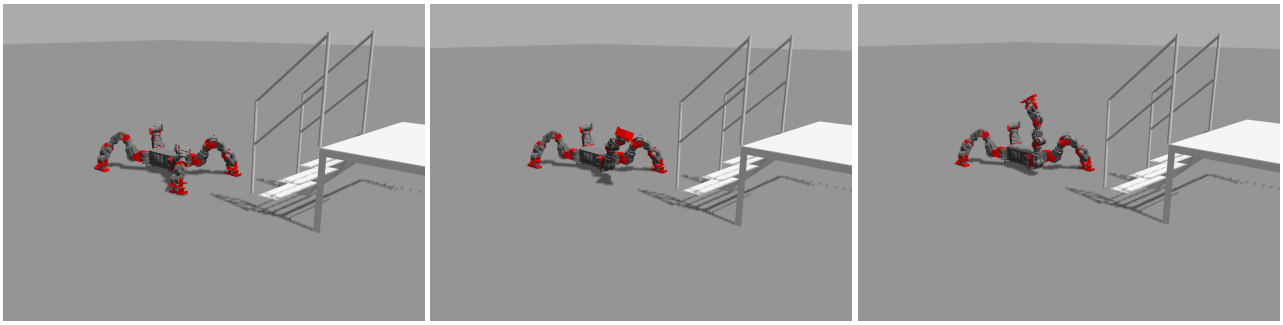


Figure 13. Image sequence of the scenario used for evaluation in (Gazebo/ROS) simulation

step:

$$SC/S = \frac{1}{T}\sum_{t=1}^{T} IG_t \tag{21}$$

All experiments were run on an Intel® i7 4712HQ CPU and the average planning run-times were used to assess our implementation efficiency and decide which sampling parameter values are suitable for online performance (see Table 9).

Resource-constrained sampling-based scheduling We first analyzed the influence of different camera and ray cast sample sizes in the trade-off between reconstruction accuracy and run-time performance. We considered $K_{\max} = 1$, camera sample size $M \in \{1; 5; 10; 15; 20\}$, and ray cast sample size $F \in \{0.01; 0.1; 1.0; 10.0\}$ in % of N_p. We will later discuss how performance changes with the choice of $K_{\max}$.

The map resolution was set to $\delta = 0.05m$. The assessed map entropy was considered within a cube of $15m$, centered in the robot base link at $t = 0$.

We compare the performance of our method to the state-of-the-art NBV planning approach of Kriegel et al. (2015) and a naive round-robin[†] scheduling algorithm Rasmussen and Trick (2008).

[†] in round-robin scheduling each resource subset is activated (i.e. scheduled) in equal portions and in circular order

Table 9 shows computation times for the scheduling algorithm and Figure 14 shows the entropy and coverage gains per iteration. On one hand, Figure 14 shows that information-aware approaches are intrinsically more exploitative than deterministic round-robin scheduling strategies, since they prefer maximum entropy never-seen regions of the map. This is useful in fast-motion scenarios such as ours, since there are short time-windows when certain map regions are visible. The figure shows that our method improves performance with respect to both a round-robin strategy and to Kriegel et al. (2015)'s method. The latter actually performs worse than baseline due to its greedy utility function which tends to get trapped in high-entropy, no-visibility viewpoints, as exhibited by the map entropy curve in Figure 14 that saturates after a few iterations. Our proposed UCB1 utility combined with anticipated reconstruction improvements, ensures each camera is selected infinitely often, hence promoting activation switches from time to time and higher long-term payoffs independent of the reconstruction scenario.

As can be seen in Table 6, Table 7 and Figure 14, carefully balancing the planner sample size (i.e. number of considered cameras and ray casts) can improve average performance. Smaller sampling sizes result in faster planning and thus more reactive exploratory behaviour (i.e. higher camera acquisition frequency), while higher sampling sizes result in better entropy approximations and rewards. However, the slower reaction of highly-sampled planning can actually lead to lower information gains per step as seen in Table 6, where for example $F = 1\%$ is better than both $F = 100\%$ and $F = 0.01\%$. This is again because the time-window for seeing certain map regions is short due to fast robot motion, and long planning times will thus lead to activating those cameras too late.

Then, we analyzed the trade-off between the the number of camera activations ($K_{\max}$) and reconstruction performance. We considered the following camera activations per time instance $K_{\max} \in \{1; 5; 10; 15; 20\}$. We used $M = 20$ and $F = 1\%$ for high performance and low computational cost based on the results in Table 9 and 6. We measured run-times for sensor acquisition and map insertion for different $K_{\max}$, which we show in Table 8. As seen in the table, these run-times increase approximately linearly with $K_{\max}$. Planning time for our method is constant and equal to 0.66 seconds, as we already saw in Table 9 ($M = 20$ and $F = 1\%$), which means that the computation overhead due to acquisition and insertion actually dominates planning time for $K_{\max} > 1$. This is reflected in a reduction of reconstruction quality per time, as we show in Figure 19 where we compare entropy and coverage gains over time using different values of $K_{\max}$. Such performance reductions could be alleviated with smarter parallel-insertion systems[‡], although this is out of scope of this paper. Because of these results, we use $K_{\max} = 1$ for the rest of the experiments.

Task-dependent sensor scheduling To analyze the influence of different camera sampling weights on mapping performance, we considered three different priors, all with standard deviation $\mathbf{\Sigma}^{n_g}$ equal in all directions (see Fig. 16):

- uniform: an unbiased distribution, created from a single Gaussian ($N_g = 1$) with zero mean, representing the absence of knowledge regarding the walking pattern and the terrain characteristics;
- ground-biased: a distribution biased towards the ground, created from a single Gaussian distribution ($N_g = 1$) with non-zero mean, larger in the bottom direction, to prioritize awareness towards the ground;
- bottom-front-biased: created from a two component GMM ($N_g = 2$) with equal weights, the first having the same statistics of the ground-biased, and the second a larger mean in the frontal direction.

The results depicted in Fig. 17 demonstrate the advantage of incorporating prior knowledge in the camera sampling procedure, when it is available. For this particular scenario, the bottom-front-biased distribution is the best performing one, since sampling is prioritized towards the bottom of the robot (ground) and the front (stairs), hence reducing planning energy/time on cameras pointing towards unoccupied areas. This result demonstrates how our method conveniently allows setting task-related scheduling priors when they are available, though they could potentially be learned. Throughout the rest of the paper we will assume the use of a uniform prior for simplicity.

Evaluation of sensor placement design in terms of mapping performance In the final simulation experiment, we once again evaluate our sensor placement design. This time we do this evaluation in terms of mapping performance instead of visual coverage (which was done in Tables 2 and 3). We compare the map entropy and occupancy obtained with our sensor placement design to the performance obtained with the alternative design approaches of before (i.e. the designs based on the state-of-the-art robots robosimian, spotmini and e2dr). The results are depicted in Fig. 18 and Tables 10 and 11. Consistently with the

[‡]We use Wurm et al. (2010)'s implementation, which is serial.

Table 6. Average entropy per sensor scheduling step (IG/S) for $K_{\max} = 1$

	$M = 20$	$M = 15$	$M = 10$	$M = 5$	$M = 1$
F=10%	72.2×10^6	72.2×10^6	72.2×10^6	72.3×10^6	72.3×10^6
F=1%	73.1×10^6	73.1×10^6	73.1×10^6	73.1×10^6	73.1×10^6
F=0.1%	73.0×10^6	73.0×10^6	73.0×10^6	73.1×10^6	73.1×10^6
F=0.01%	72.9×10^6	73.0×10^6	73.0×10^6	73.1×10^6	73.0×10^6
round-robin	72.8×10^6				
Kriegel et al. ($M = 15, F = 10\%$)	72.9×10^6				

Table 7. Average surface coverage per sensor scheduling step (SC/S) for $K_{\max} = 1$

	$M = 20$	$M = 15$	$M = 10$	$M = 5$	$M = 1$
F=10%	98.2×10^3	10.3×10^4	94.5×10^3	91.1×10^3	87.1×10^3
F=1%	95.8×10^3	91.8×10^3	86.5×10^3	81.4×10^3	80.3×10^3
F=0.1%	96.7×10^3	87.4×10^3	88.5×10^3	90.0×10^3	92.0×10^3
F=0.01%	93.1×10^3	80.8×10^3	54.7×10^3	47.5×10^3	21.4×10^3
round-robin	93.8×10^3				
Kriegel et al. ($M = 15, F = 10\%$)	96.4×10^3				

Table 8. Average sensor acquisition + map insertion run-times [s]

	$K_{\max} = 20$	$K_{\max} = 15$	$K_{\max} = 10$	$K_{\max} = 5$	$K_{\max} = 1$
Our	9.01	7.07	4.93	2.20	0.47
round-robin	9.03	6.70	4.53	2.23	0.43
Kriegel et al.	9.11	7.17	4.61	2.18	0.46

Table 9. Average planning run-times [s]

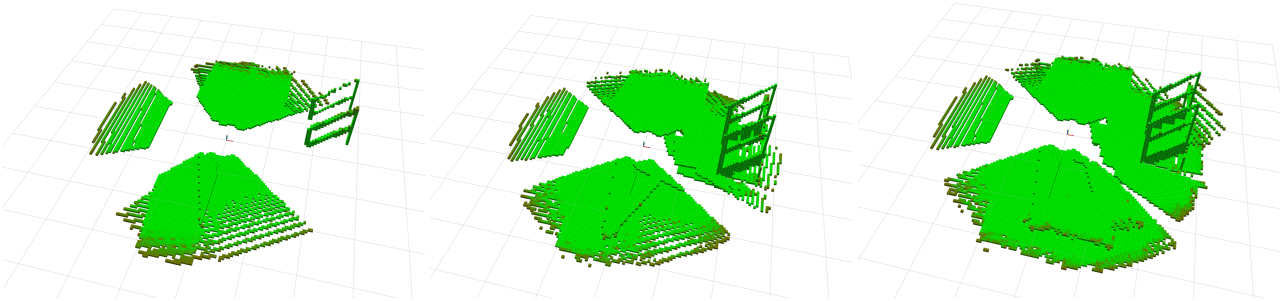
	$M = 20$	$M = 15$	$M = 10$	$M = 5$	$M = 1$
F=100%	23.18	16.98	10.89	5.26	0.99
F=10%	2.34	1.76	1.18	0.59	0.12
F=1%	0.24	0.18	0.12	0.06	0.02
F=0.1%	0.01	0.01	0.01	0.01	0.01
F=0.01%	0.01	0.01	0.01	0.01	0.01
round-robin	≈ 0				
Kriegel et al. ($M = 15, F = 10\%$)	1.6				

results of the previous section, our sensor placement design outperformed all the others, even those comprising more cameras such as the e2dr-dbl. These results show that the high coverage obtained by the design optimization algorithm also translates into high mapping quality. Importantly, they also show that our decision to decouple the many-sensor design-and-scheduling problem into a design algorithm optimizing coverage and a scheduling algorithm optimizing map entropy is effective at obtained high coverage and mapping performance.

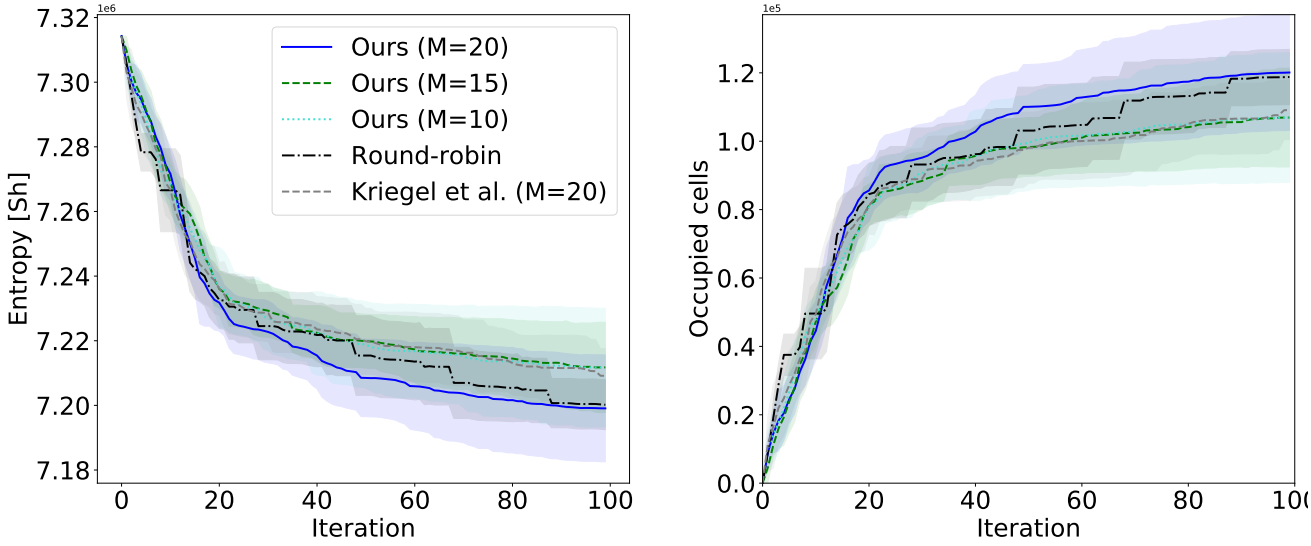
Evaluation on the real robot

With the aim of assessing the performance of the complete system on the real robot WAREC-1, we created 3 different experimental setups:

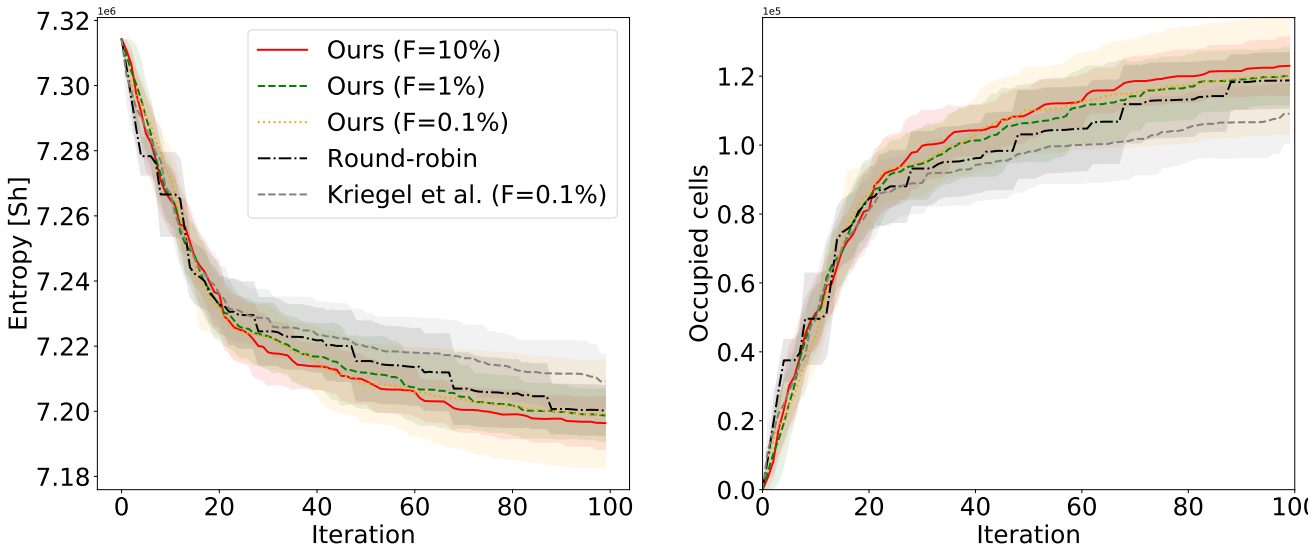
- Environment-scanning scenario (same as in the previous section): the robot lays on the floor on a crawling posture, making it too low to be able to sense the surfaces of stairs which lie in front of it. The robot then executes an up-and-down cyclical motion with one of the arms such as to perceive the surfaces from above using the sensors attached to the leg;
- Surprise-valve scenario: the robot walks with a quadruped gait using a contact and full-body planner



(a) Reconstruction evolution over sensor scheduling steps with our method for fixed $F = 10\%$ and $M = 15$



(b) Fixed $K_{\text{max}} = 1$, Fixed $F = 0.1\%$, Varying M

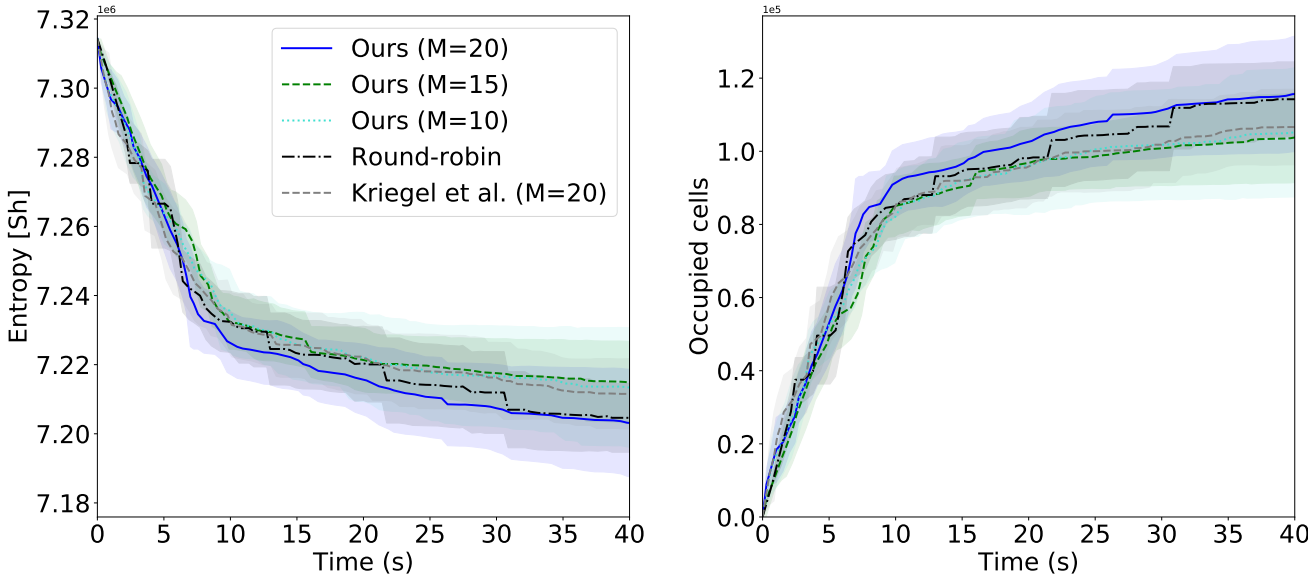


(c) Fixed $K_{\text{max}} = 1$, Fixed $M = 20$, Varying F

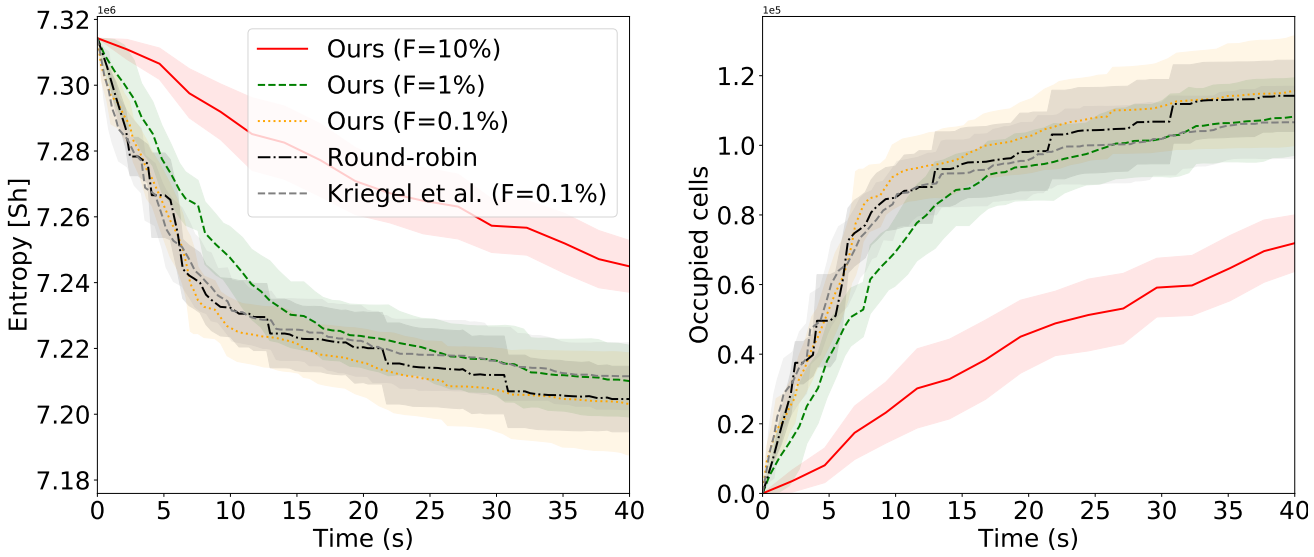
Figure 14. Mapping performance of the different scheduling algorithms in simulation, per iteration. Iteration means sensor acquisition and point cloud insertion into the occupancy grid.

Table 10. Average entropy per sensor scheduling step (IG/S) for $K_{\text{max}} = 1$ and different designs

	ours (f_n=20)	robosimian (f_n=10)	robosimian-dbl (f_n=18)	spotmini (f_n=7)	spotmini-new (f_n=5)	e2dr (f_n=22)	e2dr-dbl (f_n=44)
round-robin	723.8×10^6	729.3×10^6	729.4×10^6	729.4×10^6	727.8×10^6	730.9×10^6	731.0×10^6

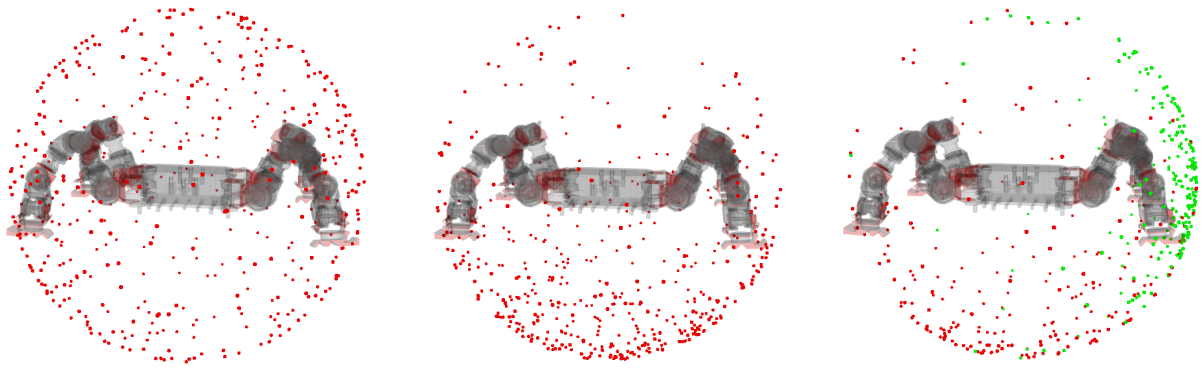


(a) Fixed $K_{\text{max}} = 1$, Fixed $F = 0.1\%$, Varying M



(b) Fixed $K_{\text{max}} = 1$, Fixed $M = 20$, Varying F

Figure 15. Temporal mapping performance of the different scheduling algorithms in simulation



(a) Uniform **(b)** Bottom biased **(c)** Bottom (red) + Front (green) biased

Figure 16. Different considered egocentric camera sampling directions.

adapted from Brandao et al. (2016). Approximately at the middle of the experiment, the space behind a wall suddenly becomes visible, revealing a valve of high mission importance;

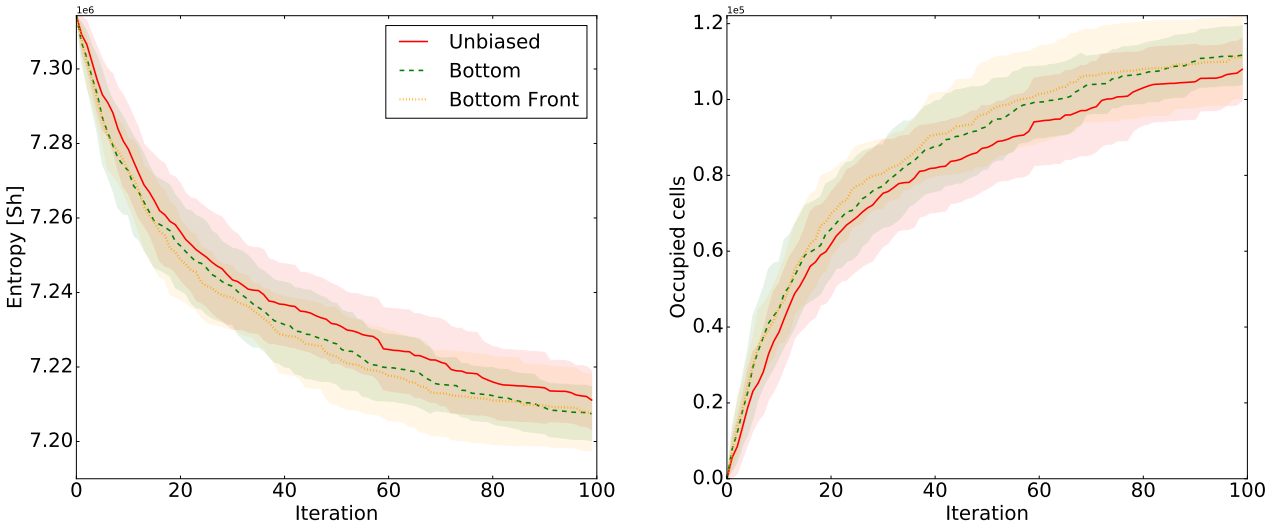


Figure 17. Mapping performance for different scheduler biases. Fixed $M = 5$

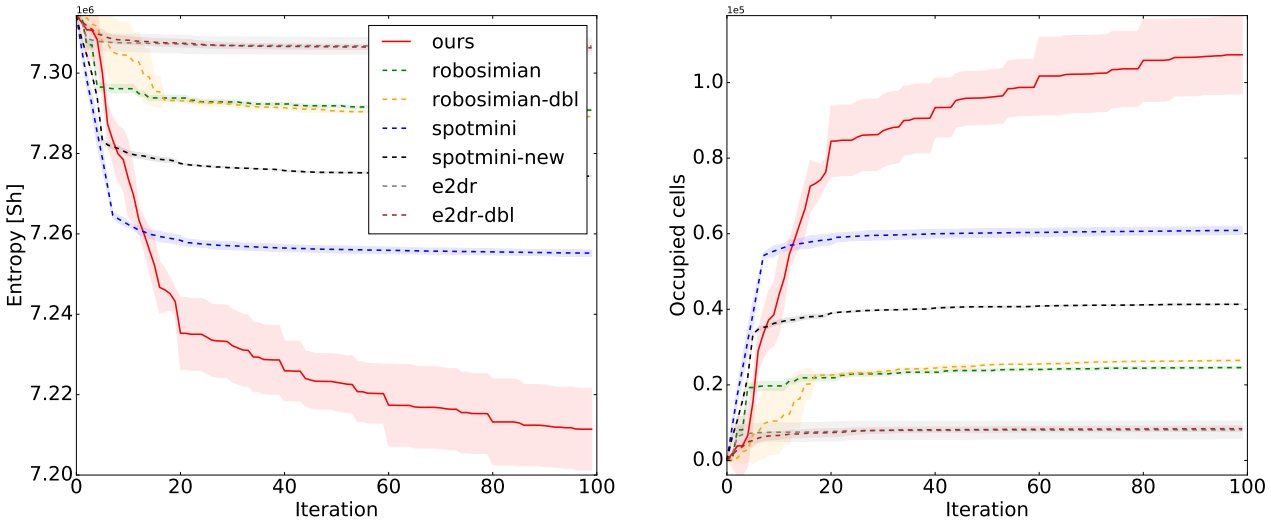


Figure 18. Mapping performance on different design approaches.

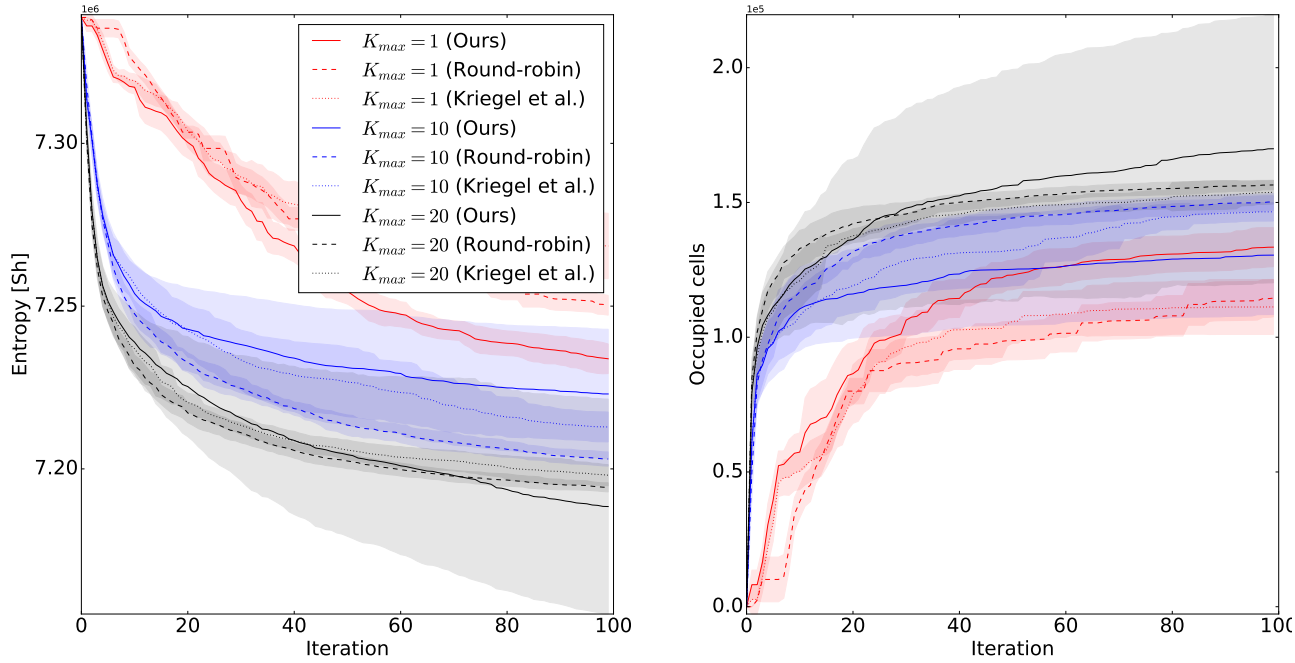
Table 11. Average surface coverage per sensor scheduling step (SC/S) for $K_{\text{max}} = 1$ and different designs

	ours (f_n=20)	robosimian (f_n=10)	robosimian-dbl (f_n=18)	spotmini (f_n=7)	spotmini-new (f_n=5)	e2dr (f_n=22)	e2dr-dbl (f_n=44)
round-robin	81.7×10^3	22.3×10^3	21.2×10^3	56.5×10^3	38.3×10^3	5.2×10^3	6.3×10^3

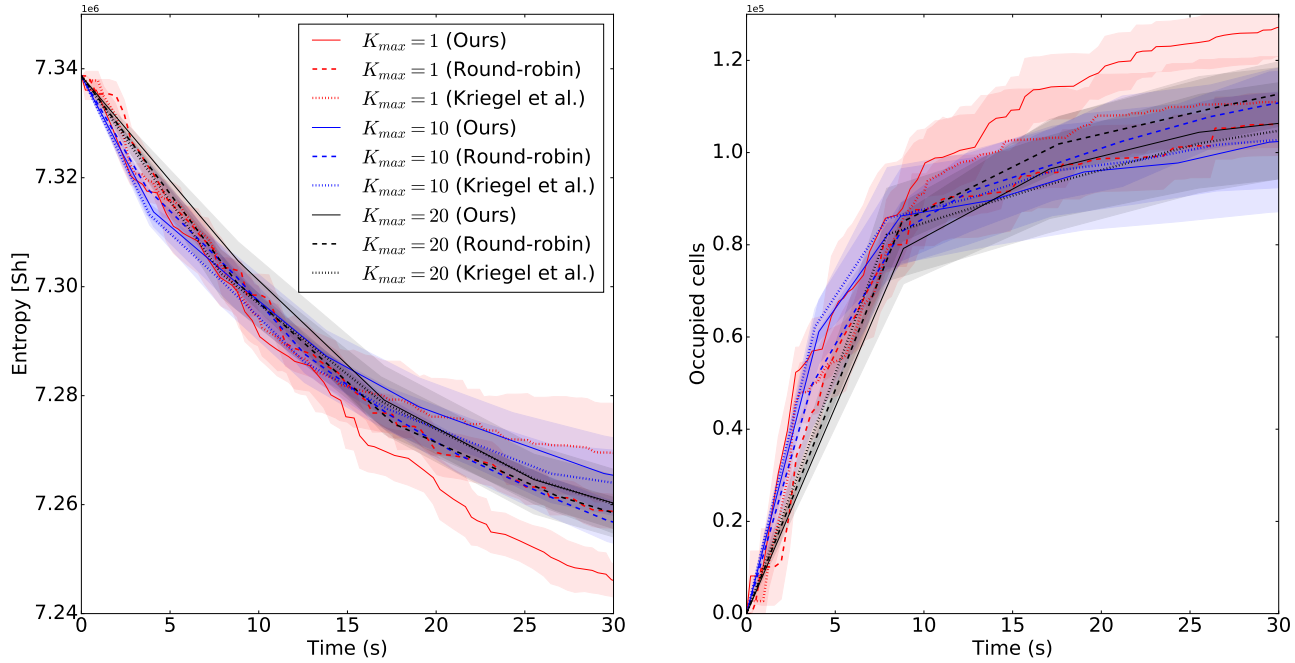
- Stair-climbing scenario: the robot climbs up a set of stairs until it reaches a high platform with rubble. It does so on a quadruped gait also planned using the same algorithm of Brandao et al. (2016). Some cameras are occluded by the robot's limbs during locomotion and should thus be avoided more often during scheduling.

Figure 20 shows image sequences of all scenarios. The test-field is a mock-up which we use for preparations for the final test-field of the Japanese ImPACT Tough Robotics Challenge Project, and is supposed to simulate search and rescue missions in indoors disaster situations (e.g. a power plant).

In Figure 21 we quantitatively assess the performance of our method against the previously mentioned baselines. In all scenarios, our method is the highest performing both in terms of information gathering and surface coverage. Both the speed of improvement and the final value of information and coverage are higher when using our method, which means that scheduling speed can also be reduced to obtain the same mapping performance and lower CPU use.



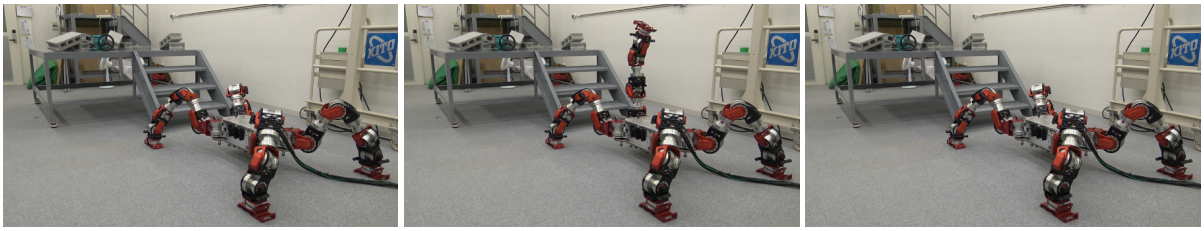
(a) Mapping performance over sensor scheduling iterations.



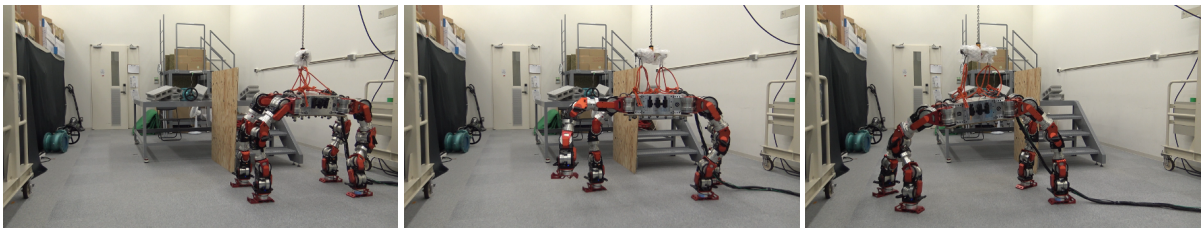
(b) Mapping performance over time.

Figure 19. Results for our and baseline sensor scheduling methods in simulation for varying K_{max}, fixed $M = 20$ and fixed $F = 1\%$.

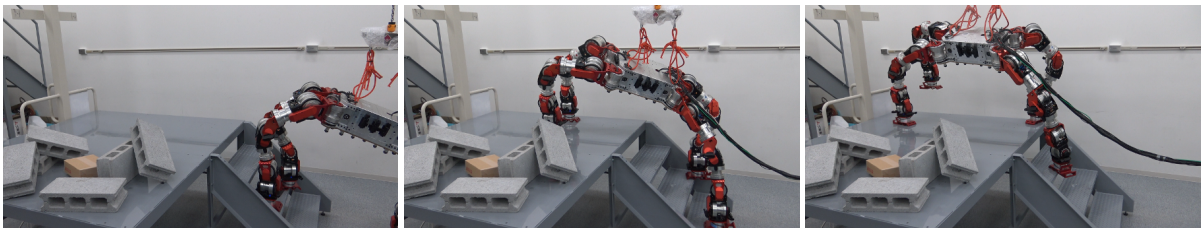
The actual maps obtained during the surprise-valve scenario intuitively demonstrate how our method leads to efficient use of computational resources in practice. Figure 22 shows these maps as the robot walks past the valve, which is circled in red. As soon as the robot walked past the wall (t=11s), the method picked the sensor which pointed at the previously-occluded region of space, thus revealing the valve. The round-robin method did so 4s later, at t=15s. This scenario qualitatively shows that our method can obtain good mapping performance compared to information-agnostic schedulers, even at low CPU loads. In this case occupancy grid insertion ran at 1Hz, while still perceiving objects quickly. Note that an agnostic scheduler such as round-robin needs to "get lucky" in order to select the right sensor once the valve becomes visible. For example, to guarantee that the valve is seen within 1 second of our



(a) Environment-scanning scenario



(b) Surprise-valve scenario



(c) Stair-climbing scenario

Figure 20. Image sequences of the scenarios used for evaluation on the real robot.

method, round-robin acquisition and grid insertion would need to run at 20hz (to run through all sensors at each second).

In these experiments all processes were ROS nodes, i.e. 20 cloud publisher (acquisition) nodes, 20 self_filter nodes and one octomap node. At the time of the experiment, the implementation of our cloud publisher nodes did not take advantage of the fact that only one cloud was being processed per second, and therefore all the cloud publishing nodes were constantly active at 45hz, spending 60% of CPU. However, in principle the implementation can be changed such as to only produce clouds on request, which would lead to approximately freeing more than 54% CPU usage. We plan to implement this change in the future. In addition to CPU usage due to acquisition, filtering used 4% (instead of 40% due to one sensor being used at a time), mapping 6% and scheduling 8% (round-robin used 7%).

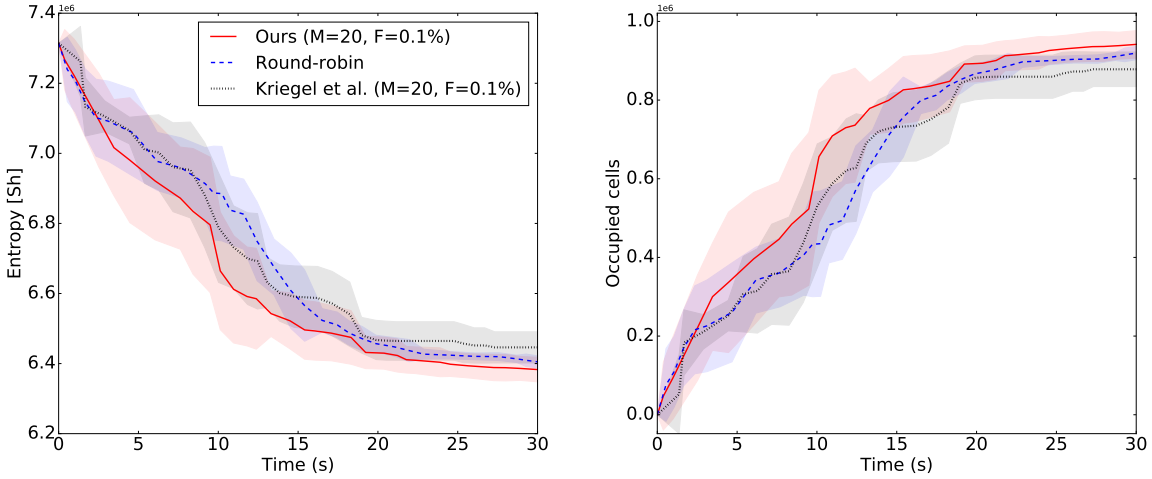
Conclusion

This paper introduced methods for both design and resource management of perception systems consisting of a large number of depth sensors. Our design optimization method obtains optimal trade-offs of spherical coverage and the number of sensors, and includes task-specific coverage constraints, symmetry constraints and considerations of the distribution of robot motion. Our resource management method schedules sensor processing on a mapping task by selecting which sensors to use at which time. It does so based on a NBV planning approach with probabilistic sensor models, information gain metrics and sampling techniques for high-quality mapping with low CPU consumption.

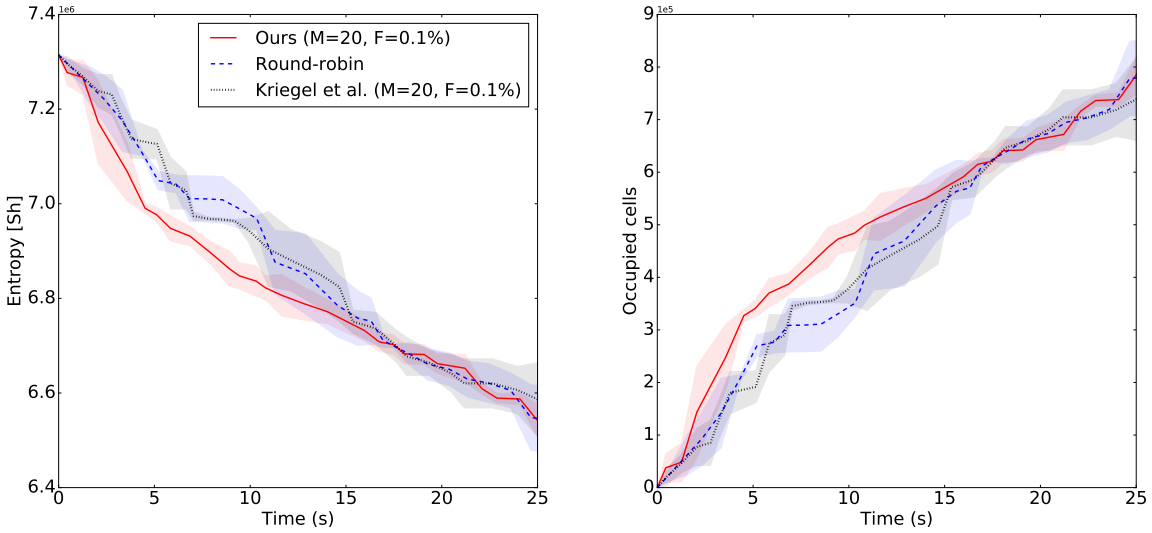
We showed through comprehensive simulation experiments that the design optimization method is capable of obtaining up to 80% visual coverage on sample locomotion tasks using 29 sensors, while design approaches of state-of-the-art robotic platforms achieve only around 60%. Our final design uses 20 sensors, has both high spherical coverage of 65% and complete (100%) coverage of end-effector positions. We also evaluated the whole system on the real WAREC-1 robot, both quantitatively and qualitatively. The system successfully maps environments faster than naive round-robin scheduling as well as the state-of-the-art information-based NBV planning approach of Kriegel et al. (2015). Furthermore, the system can be pushed to 45Hz update rates if enough processing power is available.

Discussion

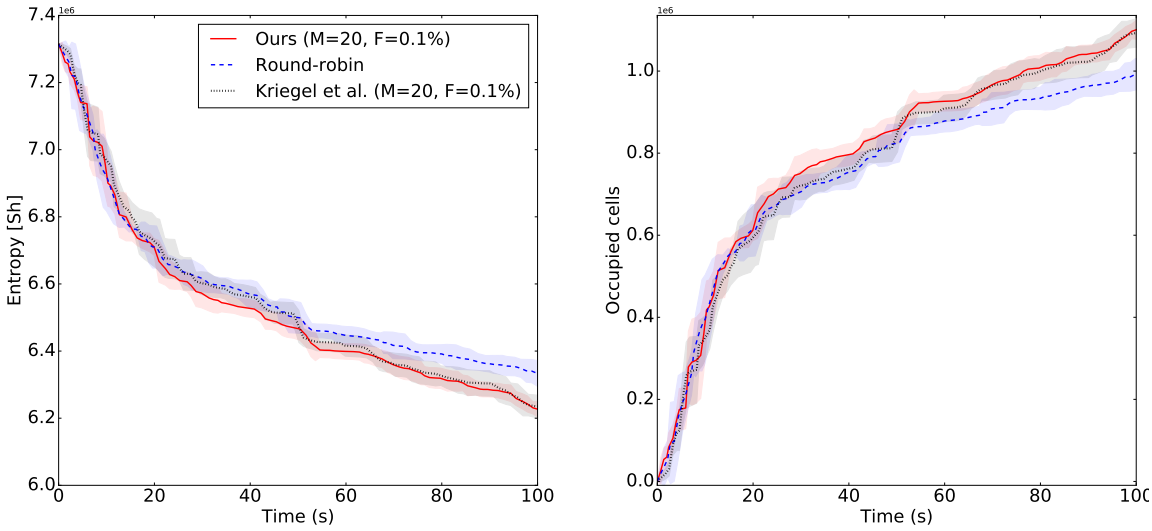
We strongly believe that wide-coverage and active vision approaches to sensing such as those introduced in this paper are of utmost importance for robots deployed in the real world, where contact surfaces and obstacles might



(a) Environment-scanning scenario



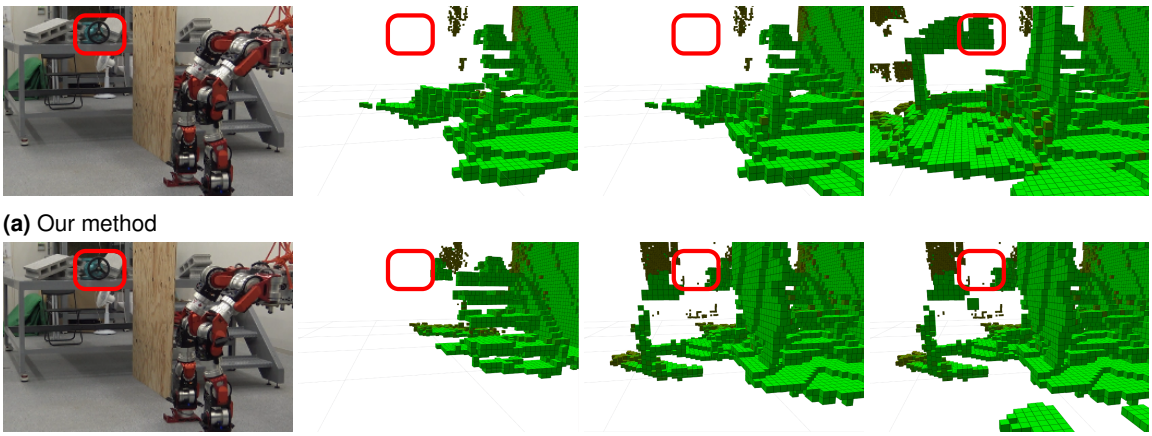
(b) Surprise-valve scenario



(c) Stair-climbing scenario

Figure 21. Quantitative results for ours and baseline methods in a real environment

be complex and thus they should be frequently sensed for autonomous re-planning tasks, teleoperation tasks, and others. In addition to that, since there is only so much computing power that can be placed inside the robot, we



(a) Our method

(b) Round-robin

Figure 22. Evolution of the occupancy grid over time (t=3,7,11s) on the surprise-valve scenario. The red rectangle marks the object of interest, which hidden behind a wooden wall at t=0. When walking past the wooden wall, our method quickly picks the previously-occluded viewing direction which reveals a valve (at t=11s) before the round-robin method does (at t=15s).

feel that processing resources should be managed such as to provide effective active exploration, and autonomously such as to reduce the burden on human operators. For example, even though our system uses 2 PCs for perception, acquiring and filtering all point clouds simultaneously already exhausts the whole CPU power of both computers: 60% usage for acquisition, and an extra 40% usage for filtering out points which lie on the robot's links (average 6 and 4% per camera respectively). Our scheduling algorithm is flexible enough that you can specify the number of sensors to process and useful enough that it can focus on the right regions of space when doing so.

There is a number of lessons that we learned from the research presented here, which we would like researchers and practitioners to take into consideration when building on our work:

Blind-spots and objective specification While the final design we implemented on the real robot had optimal coverage for the number of sensors used, and planned end-effector positions were visible from previous configurations, we later realized that important parts of the working space were not covered. Specifically, front and back regions of the locomotion plane during quadrupedal walking are not visible except at contact points, which in practice makes it difficult to plan from the obtained maps unless the robot walks sideways in quadrupedal mode. While walking sideways is not a problem for our robot in terms of capabilities, this limitation was not necessary. The lesson we learned is that objective specification is difficult even within the Pareto-front of coverage and sensor-count. In retrospect, we should have simulated the different locomotion modes in a robot simulator (e.g. Gazebo) to analyse whether we were satisfied with the (un)covered regions in each mode - before jumping to implementation on the real-robot. In other words, we trusted too much on the sufficiency of our task-specific constraint, while there were also other important constraints. We also weighted the cost of installing and buying extra sensors too heavily, and picking a solution with around 2 or 4 extra sensors from the Pareto-front would have alleviated this particular blind-spot problem. In the particular case of our robot, our plan is to now keep these sensor placements and re-run the optimization algorithm with free extra sensors to increase coverage. Additionally, we plan to more intensively evaluate performance in simulated tasks before real implementation.

Specification-optimization loop More generally, and following on the previous point, we believe that in practice the design of many-sensor systems through optimization, as we propose here, should be made in a loop of objective specification, optimization, manually and qualitatively inspecting performance of the Pareto solutions, back to specification, etc. This is to further evaluate our actual needs and help us through the process of understanding what we as designers and robot users really want from the robot. This problem of (mis)specification is actually an important and trending topic of research, in particular in the artificial intelligence community of safety Amodei et al. (2016), and it pervades many robotics, planning and learning problems. In the specific case of our algorithm, "specification" includes both the specification of the objectives (e.g. spherical and contact coverage) and the specification of the set of tasks to optimize for. Additionally, manually added sensor locations, the user-definition of sets of links or faces where sensors cannot be attached, and the discretization used for the automatic placement of sensors, are all also different kinds of specification that will affect the final design. Misjudgements

may lead to designs that end up not being feasible in terms of assembly, or to not placing sensors where actually this was possible and optimal. These considerations will also be part of the specification-optimization loop in our problem.

Research on coverage definitions We understood that there is a need for research on other definitions of coverage other than "percentage of directions/points covered" (expected value in our case). On the one hand, we want to experiment with the definition of task-specific coverage, such that it includes whole regions important for locomotion and not just end-effector points (e.g. locomotion plane, ladder rungs, etc), as well as more complex environment priors. But more importantly, maximizing the expected number of covered points ignores how these points are actually distributed. We believe there is a need to research and evaluate other metrics which also include distribution considerations, for example metrics of uniformity, the maximum area of blindspots, and others.

Sensor modalities and redundancy The particular sensor we use is based on infrared but still has reasonable performance outdoors. However, it has problems perceiving many surface materials: from very bright, to transparent and shiny-metal surfaces. This fact is problematic not only for perception algorithms in general, but also for information-based scheduling since unseen regions of "bad" materials may be sampled more frequently than others even if nothing is measured. At the scheduling level, research on alternatives to our UCB term such as inhibition-of-return methods or methods for adaptively changing priors for each sensor could reveal fruitful. At the design side, ideally the optimization should be made with a set of complementary sensors (e.g. infrared, LIDAR, stereo) to obtain redundancy and higher robustness to environment conditions.

Task priors for scheduling The task being executed can actually provide many priors for how to effectively sample cameras, independently of the planning horizon, while assigning more weight to specific cameras. For example, the locomotion direction, low-confidence areas close to collision, the type of terrain, etc. In this work, we proposed to use a randomized sampling mechanism to egocentrically encode task-dependent priorities on sensor orientations, using Gaussian Mixture Models (GMMs). Cameras pointing towards higher density regions in the unit sphere of orientations, are more likely to be selected at run-time. In the future, it would be interesting to learn these priors directly from experience, as well as to develop ways of automatically detecting task switches.

Computation overheads In some cases, planning agnostic solutions (i.e. round-robin or random) might perform better per time instant due to the computation time overhead of planning. While the planning time can be made negligible and still keep high performance, it could become significant at very high sampling frequencies (e.g. with high-spec PCs or more on-board PC units). The overhead can nevertheless be reduced since our algorithm is highly paralelizable. For instance, the planning algorithm could be easily implemented on GPU by parallelizing the reward computation, based on independent summations. Finally, our current implementation relies on data acquisition from the selected resource only after decision, which introduces another time overhead depending on data availability and throughput. One could for instance benefit from a dedicated buffered data acquisition thread, running in parallel.

Non-myopic planning Although the proposed sensor scheduling optimization formulation allows considering arbitrary planning horizons, the main goal during the evaluation of our scheduling approach was assessing the trade-off between mapping reconstruction performance and resource availability (i.e. planning and mapping timings), in a multi-camera system, mounted on a resource-constrained multiple degree of freedom robot. Therefore, we only assessed one step ahead greedy predictions. For future work, one may consider the use of look-ahead tree search techniques, namely beam search Ortmanns and Ney (2000), or Monte-Carlo Tree Search (MCTS) based approaches Kocsis and Szepesvári (2006); Browne et al. (2012). These tackle the planning complexity's exponential growth with the time horizon, by expanding only the most promising nodes at each planning tree level.

Acknowledgments

This work was supported by ImPACT TRC Program of Council for Science, Technology and Innovation (Cabinet Office, Government of Japan). Martim Brandao was partially funded by the UKRI/EPSRC ORCA Hub [EP/R026173/1]. Rui Figueiredo was funded by the Portuguese Foundation for Science and Technology (FCT) project [UID/EEA/50009/2019].

The authors would also like to thank Takashi Matsuzawa for his advice on hardware design, and Yuki Yoshida, Nobuaki Sakai, Masahiro Okawara, Kengo Kumagai and Keisuke Namura for their help with experiments.

References

(2005) Open dynamics engine. URL http://www.ode.org/.

Agrawal R (1995) Sample mean based index policies with o (log n) regret for the multi-armed bandit problem. *Advances in Applied Probability* : 1054–1078.

Ahmad S and Yu AJ (2013) Active sensing as bayes-optimal sequential decision making. *CoRR* abs/1305.6650. URL http://arxiv.org/abs/1305.6650.

Aloimonos J, Weiss I and Bandyopadhyay A (1988) Active vision. *International journal of computer vision* 1(4): 333–356.

Amodei D, Olah C, Steinhardt J, Christiano P, Schulman J and Mané D (2016) Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565* .

Atkeson CG, Babu BPW, Banerjee N, Berenson D, Bove CP, Cui X, DeDonato M, Du R, Feng S, Franklin P, Gennert M, Graff JP, He P, Jaeger A, Kim J, Knoedler K, Li L, Liu C, Long X, Padir T, Polido F, Tighe GG and Xinjilefu X (2015) No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. pp. 623–630. DOI:10.1109/HUMANOIDS.2015.7363436.

Auer P, Cesa-Bianchi N and Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3): 235–256.

Bircher A, Kamel M, Alexis K, Oleynikova H and Siegwart R (2016) Receding horizon" next-best-view" planner for 3d exploration. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 1462–1468.

Bodor R, Schrater P and Papanikolopoulos N (2005) Multi-camera positioning to optimize task observability. In: *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.* pp. 552–557. DOI:10.1109/AVSS.2005.1577328.

Brandao M, Ferreira R, Hashimoto K, Santos-Victor J and Takanishi A (2013) Active gaze strategy for reducing map uncertainty along a path. In: *3rd IFToMM International Symposium on Robotics and Mechatronics*. pp. 455–466.

Brandao M, Shiguematsu YM, Hashimoto K and Takanishi A (2016) Material recognition cnns and hierarchical planning for biped robot locomotion on slippery terrain. In: *16th IEEE-RAS International Conference on Humanoid Robots*. pp. 81–88. DOI:10.1109/HUMANOIDS.2016.7803258.

Brodbeck L, Hauser S and Iida F (2018) *Robotic Invention: Challenges and Perspectives for Model-Free Design Optimization of Dynamic Locomotion Robots*. Cham: Springer International Publishing. ISBN 978-3-319-60916-4, pp. 581–596. DOI: 10.1007/978-3-319-60916-4_33.

Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S and Colton S (2012) A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1): 1–43.

Butko NJ and Movellan JR (2010) Infomax control of eye movements. *Autonomous Mental Development, IEEE Transactions on* 2(2): 91–107.

Chakrabarty K, Iyengar SS, Qi H and Cho E (2002) Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers* 51(12): 1448–1453. DOI:10.1109/TC.2002.1146711.

Chen S, Li Y and Kwok NM (2011) Active vision in robotic systems: A survey of recent developments. *International Journal of Robotics Research* 30(11): 1343–1377.

Chen SY and Li YF (2004) Automatic sensor placement for model-based robot vision. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34(1): 393–408. DOI: 10.1109/TSMCB.2003.817031.

Cheney N, MacCurdy R, Clune J and Lipson H (2013) Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13. New York, NY, USA: ACM. ISBN 978-1-4503-1963-8, pp. 167–174. DOI:10.1145/2463372.2463404.

Chong EK, Kreucher CM and Hero AO (2008) Monte-carlo-based partially observable markov decision process approximations for adaptive sensing. In: *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*. IEEE, pp. 173–180.

Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering* 191(11): 1245 – 1287. DOI:10.1016/S0045-7825(01)00323-1.

de Figueiredo RP, Bernardino A, Santos-Victor J and Araújo H (2018) On the advantages of foveal mechanisms for active stereo systems in visual search tasks. *Autonomous Robots* 42(2): 459–476.

Deb K, Pratap A, Agarwal S and Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 6(2): 182–197. DOI:10.1109/4235.996017.

Delmerico J, Isler S, Sabzevari R and Scaramuzza D (2018) A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots* 42(2): 197–208.

Diankov R (2010) *Automated Construction of Robotic Manipulation Programs*. PhD Thesis, Carnegie Mellon University, Robotics Institute.

Dornhege C and Kleiner A (2013) A frontier-void-based approach for autonomous exploration in 3d. *Advanced Robotics* 27(6): 459–468.

Efraimidis PS and Spirakis PG (2006) Weighted random sampling with a reservoir. *Information Processing Letters* 97(5): 181–185.

Englot B and Hover FS (2013) Three-dimensional coverage planning for an underwater inspection robot. *The International Journal of Robotics Research* 32(9-10): 1048–1073. DOI: 10.1177/0278364913490046.

Fallon M, Kuindersma S, Karumanchi S, Antone M, Schneider T, Dai H, D'Arpino CP, Deits R, DiCicco M, Fourie D, Koolen T, Marion P, Posa M, Valenzuela A, Yu KT, Shah J, Iagnemma K, Tedrake R and Teller S (2015) An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics* 32(2): 229–254. DOI:10.1002/rob.21546. URL http://dx.doi.org/10.1002/rob.21546.

Figueiredo R, Avelino J, Dehban A, Bernardino A, Lima P and Arajo H (2017) Efficient resource allocation for sparse multiple object tracking. In: *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 6: VISAPP, (VISIGRAPP 2017)*. INSTICC, SciTePress. ISBN 978-989-758-227-1, pp. 300–307. DOI:10.5220/0006173103000307.

Fortin FA, De Rainville FM, Gardner MA, Parizeau M and Gagné C (2012) DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research* 13: 2171–2175.

Gutmann JS, Fukuchi M and Fujita M (2005) A floor and obstacle height map for 3d navigation of a humanoid robot. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, pp. 1066–1071.

Hashimoto K, Kimura S, Sakai N, Hamamoto S, Koizumi A, Sun X, Matsuzawa T, Teramachi T, Yoshida Y, Imai A, Kumagai K, Matsubara T, Yamaguchi K, Ma G, and Takanishi A (2017) Warec-1 - a four-limbed robot having high locomotion ability with versatility in locomotion styles. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. pp. 172–178.

Haynes GC, Stager D, Stentz A, Vande Weghe JM, Zajac B, Herman H, Kelly A, Meyhofer E, Anderson D, Bennington D, Brindza J, Butterworth D, Dellin C, George M, Gonzalez-Mora J, Jones M, Kini P, Laverne M, Letwin N, Perko E, Pinkston C, Rice D, Scheifflee J, Strabala K, Waldbaum M and Warner R (2017) Developing a robust disaster response robot: Chimpand the robotics challenge. *Journal of Field Robotics* 34(2): 281–304. DOI:10.1002/rob.21696. URL http://dx.doi.org/10.1002/rob.21696.

Herbert M, Caillas C, Krotkov E, Kweon IS and Kanade T (1989) Terrain mapping for a roving planetary explorer. In: *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE, pp. 997–1002.

Hold-Geoffroy Y, Gagnon O and Parizeau M (2014) Once you scoop, no need to fork. In: *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. ACM, p. 60.

Hornung A, Wurm KM, Bennewitz M, Stachniss C and Burgard W (2013) Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots* 34(3): 189–206.

Hörster E and Lienhart R (2006) On the optimal placement of multiple visual sensors. In: *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, VSSN '06. New York, NY, USA: ACM. ISBN 1-59593-496-0, pp. 111–120. DOI:10.1145/1178782.1178800. URL http://doi.acm.org/10.1145/1178782.1178800.

Hutter M, Gehring C, Lauber A, Gunther F, Bellicoso CD, Tsounis V, Fankhauser P, Diethelm R, Bachmann S, Bloesch M, Kolvenbach H, Bjelonic M, Isler L and Meyer K (2017) Anymal - toward legged robots for harsh environments. *Advanced Robotics* 31(17): 918–931. DOI:10.1080/01691864.2017.1378591.

Isler S, Sabzevari R, Delmerico J and Scaramuzza D (2016) An information gain formulation for active volumetric 3d reconstruction. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 3477–3484.

Jadidi MG, Miro JV and Dissanayake G (2018) Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots* 42(2): 273–290.

Kaneko K, Morisawa M, Kajita S, Nakaoka S, Sakaguchi T, Cisneros R and Kanehiro F (2015) Humanoid robot hrp-2kai - improvement of hrp-2 towards disaster response tasks. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. pp. 132–139. DOI:10.1109/HUMANOIDS.2015.7363526.

Karumanchi S, Edelberg K, Baldwin I, Nash J, Reid J, Bergh C, Leichty J, Carpenter K, Shekels M, Gildner M, Newill-Smith D, Carlton J, Koehler J, Dobreva T, Frost M, Hebert P, Borders J, Ma J, Douillard B, Backes P, Kennedy B, Satzinger B, Lau C, Byl K, Shankar K and Burdick J (2017) Team robosimian: Semi-autonomous mobile manipulation at the 2015 darpa robotics challenge finals. *Journal of Field Robotics* 34(2): 305–332. DOI:10.1002/rob.21676. URL http://dx.doi.org/10.1002/rob.21676.

Katehakis MN and Veinott Jr AF (1987) The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research* 12(2): 262–268.

Keyes B, Casey R, Yanco HA, Maxwell BA and Georgiev Y (2006) Camera placement and multi-camera fusion for remote robot operation. In: *Proceedings of the IEEE international workshop on safety, security and rescue robotics*. National Institute of Standards and Technology Gaithersburg, MD, pp. 22–24.

Kocsis L and Szepesvári C (2006) Bandit based monte-carlo planning. In: *European conference on machine learning*. Springer, pp. 282–293.

Koenig N and Howard A (2004) Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3. IEEE, pp. 2149–2154.

Kriegel S, Rink C, Bodenmüller T and Suppa M (2015) Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. *Journal of Real-Time Image Processing* 10(4): 611–631.

Martins JR and Lambe AB (2013) Multidisciplinary design optimization: a survey of architectures. *AIAA journal* 51(9): 2049–2075.

Mayol-Cuevas WW, Tordoff BJ and Murray DW (2009) On the choice and placement of wearable vision sensors. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 39(2): 414–425. DOI:10.1109/TSMCA.2008.2010848.

Michel P, Chestnutt J, Kuffner J and Kanade T (2005) Vision-guided humanoid footstep planning for dynamic environments. In: *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*. IEEE, pp. 13–18.

Millington I and Funge J (2016) *Artificial intelligence for games*. CRC Press.

Moravec H and Elfes A (1985) High resolution maps from wide angle sonar. In: *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2. IEEE, pp. 116–121.

Mutlu M, Melo K, Vespignani M, Bernardino A and Ijspeert AJ (2015) Where to place cameras on a snake robot: Focus on camera trajectory and motion blur. In: *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. pp. 1–8. DOI:10.1109/SSRR.2015.7442948.

Nguyen CV, Izadi S and Lovell D (2012) Modeling kinect sensor noise for improved 3d reconstruction and tracking. In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*. IEEE, pp. 524–530.

O'Callaghan S, Ramos FT and Durrant-Whyte H (2009) Contextual occupancy maps using gaussian processes. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, pp. 1054–1060.

O'Callaghan ST and Ramos FT (2012) Gaussian process occupancy maps. *The International Journal of Robotics Research* 31(1): 42–62.

Ortmanns S and Ney H (2000) Look-ahead techniques for fast beam search. *Computer Speech & Language* 14(1): 15–32.

Palazzolo E and Stachniss C (2017) Information-driven autonomous exploration for a vision-based mav. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4: 59.

Park JH and Asada H (1993) Concurrent design optimization of mechanical structure and control for high speed robots. In: *1993 American Control Conference*. pp. 2673–2679.

Pettersson M and lvander J (2009) Drive train optimization for industrial robots. *IEEE Transactions on Robotics* 25(6): 1419–1424. DOI:10.1109/TRO.2009.2028764.

Pineau J, Gordon G and Thrun S (2006) Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research* 27: 335–380.

Radford NA, Strawser P, Hambuchen K, Mehling JS, Verdeyen WK, Donnan AS, Holley J, Sanchez J, Nguyen V, Bridgwater L, Berka R, Ambrose R, Myles Markee M, Fraser-Chanpong NJ, McQuin C, Yamokoski JD, Hart S, Guo R, Parsons A, Wightman B, Dinh P, Ames B, Blakely C, Edmondson C, Sommers B, Rea R, Tobler C, Bibby H, Howard B, Niu L, Lee A, Conover M, Truong L, Reed R, Chesney D, Platt R, Johnson G, Fok CL, Paine N, Sentis L, Cousineau E, Sinnet R, Lack J, Powell M, Morris B, Ames A and Akinyode J (2015) Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics* 32(3): 397–419. DOI:10.1002/rob.21560. URL http://dx.doi.org/10.1002/rob.21560.

Rasmussen RV and Trick MA (2008) Round robin scheduling–a survey. *European Journal of Operational Research* 188(3): 617–636.

Ross S, Pineau J, Paquet S and Chaib-Draa B (2008) Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research* 32: 663–704.

Saravanan R, Ramabalan S, Ebenezer NGR and Dharmaraja C (2009) Evolutionary multi criteria design optimization of robot grippers. *Applied Soft Computing* 9(1): 159 – 172. DOI: j.asoc.2008.04.001.

Schulz A, Sung C, Spielberg A, Zhao W, Cheng R, Grinspun E, Rus D and Matusik W (2017) Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research* 36(10): 1131–1147. DOI:10.1177/0278364917723465.

Scott WR, Roth G and Rivest JF (2003) View planning for automated three-dimensional object reconstruction and

inspection. *ACM Computing Surveys (CSUR)* 35(1): 64–96.

Sims K (1994) Evolving virtual creatures. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, pp. 15–22.

Spaan MT (2012) Partially observable markov decision processes. In: *Reinforcement Learning*. Springer, pp. 387–414.

Sutton RS and Barto AG (1998) *Introduction to Reinforcement Learning*. 1st edition. Cambridge, MA, USA: MIT Press.

Thrun S, Burgard W and Fox D (2005) *Probabilistic robotics*.

Triebel R, Pfaff P and Burgard W (2006) Multi-level surface maps for outdoor terrain mapping and loop closing. In: *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, pp. 2276–2282.

Tsagarakis NG, Caldwell DG, Negrello F, Choi W, Baccelliere L, Loc V, Noorden J, Muratore L, Margan A, Cardellino A, Natale L, Mingo Hoffman E, Dallali H, Kashiri N, Malzahn J, Lee J, Kryczka P, Kanoulas D, Garabini M, Catalano M, Ferrati M, Varricchio V, Pallottino L, Pavan C, Bicchi A, Settimi A, Rocchi A and Ajoudani A (2017) Walk-man: A high-performance humanoid platform for realistic environments. *Journal of Field Robotics* 34(7): 1225–1259. DOI:10.1002/rob.21702. URL http://dx.doi.org/10.1002/rob.21702.

van der Leeuw M (2009) The playstation 3's spu's in the real world - a killzone 2 case study. URL https://www.gdcvault.com/play/1331/The-PlayStation-3-s-SPU.

Wurm KM, Hornung A, Bennewitz M, Stachniss C and Burgard W (2010) Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In: *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2.

Yamauchi B (1997) A frontier-based approach for autonomous exploration. In: *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. pp. 146–151. DOI:10.1109/CIRA.1997.613851.

Yoshiike T, Kuroda M, Ujino R, Kaneko H, Higuchi H, Iwasaki S, Kanemoto Y, Asatani M and Koshiishi T (2017) Development of experimental legged robot for inspection and disaster response in plants. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 4869–4876. DOI:10.1109/IROS.2017.8206364.

Zitzler E, Laumanns M and Thiele L (2001) Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report* 103.