



DR 4.4: Implementing persistence in collaborative planning

Luigi Freda*, Fiora Pirri*, Tomáš Svoboda[‡], Martin Pecka[‡], Abel Gawel[‡], Renaud Dubé[‡], Cesar Cadena[‡], and the TRADR Consortium

**Alcor Laboratory, Department of Computer, Control, and Management Engineering “Antonio Ruberti” - Sapienza University of Rome.*

[†]ETH Zürich - Autonomous Systems Lab, Zurich, Switzerland

[‡]CTU - Czech Technical University in Prague, Czech Republic

`<{freda,pirri}@dis.uniroma1.it>`

<i>Project, project Id:</i>	EU FP7 TRADR / ICT-60963
<i>Project start date:</i>	Nov 1 2013 (50 months)
<i>Due date of deliverable:</i>	M38
<i>Actual submission date:</i>	February 2016
<i>Lead partner:</i>	ROMA
<i>Revision:</i>	final
<i>Dissemination level:</i>	PU

This document describes the progress status of the research work of WP4 in Year 4 of TRADR project.

WP4 developed a framework for multi-robot path planning, patrolling, exploration and coverage. The system is distributed and is based on a two level coordination strategy.

The different capabilities of the system allows to attain persistence in all the different phases of a typical long-term mission. Exploration is used to build and save a 3D map of an unknown environment. Patrolling allows to build and save a patrolling graph, i.e. a topological map representing a set of point of interests connected by safe paths. Then, the robot team can be called to *(i)* patrol the assigned points of interest by moving over the patrolling graph or *(ii)* perform a coverage task with the goal of revisiting the whole environment and update the saved 3D map.

The document reports both the research and engineering work that has been performed, in collaboration with other project partners, to effectively deploy the developed multi-robot system in real scenarios.

1	Tasks, objectives, results	8
1.1	Planned work	8
1.2	Addressing reviewers' comments	8
1.3	Actual work performed	9
1.3.1	3D Multi-robot Patrolling	9
1.3.2	3D Multi-robot Exploration	19
1.3.3	Integration of Adaptive Traversal Function in Path Planning	31
1.3.4	Point Cloud Segmentation and Clustering for Traversability Analysis	33
1.3.5	3D Reconstruction and Incremental Segmentation by using RGBD-SLAM	34
1.3.6	Software Development and Release	39
1.4	Relation to the state-of-the-art	40
1.4.1	Multi-robot patrolling	41
1.4.2	Multi-robot Exploration and Coverage	42
1.4.3	Efficient Multi-DOF Path Planning	45
2	Annexes	57
2.1	Freda (2017), "3D Multi-Robot Patrolling with a Two-Level Coordination Strategy: Simulations and Experiments"	57
2.2	Freda (2018), "3D Multi-Robot Exploration with a Two Level Coordination Strategy and Prioritization"	58
2.3	Freda (2017), "3D Multi-Robot Exploration and Coverage with a Receding Horizon Next-Best-View Approach"	58
2.4	Freda (2017), "3D Multi-Robot Patrolling with a Two-Level Coordination Strategy"	59
2.5	Freda (2018), "PLVS: An Open-Source RGB-D SLAM System with Key-points, Keylines, Volumetric Mapping and 3D Incremental Segmentation"	59
2.6	Brandizzi (2017), "Comparison between Segmentation Methods for Point Clouds"	60
2.7	Wiki-MR-Use-Cases (2017), "Review Yr3 Recommendations for WP4"	61
2.8	Wiki-MR-Use-Cases (2017), "Multi-Robot Use Cases Wiki page on Redmine"	62
2.9	Wiki-VREP (2017), "V-REP Simulation Wiki page on Redmine"	62
2.10	Wiki-MR (2017), "Multi-Robot Path Planning and Patrolling Wiki page on Redmine"	63
A	TRADR Techday Exploration poster	64
B	TRADR Techday Patrolling poster	65
C	Review Year 3 Recommendations for WP4 page on Redmine	66
D	Multi-Robot Use Cases Wiki page on Redmine	69
E	V-REP Simulation Wiki page on Redmine	76
F	Multi-Robot Path Planning and Patrolling Wiki page on Redmine	79

Executive Summary

This report describes the research work of WP4 toward the development and implementation of models of persistence in multi-robot collaboration.

In Year 4, the main result of WP4 is the development of a framework for multi-robot path planning, patrolling, exploration and coverage. Behind this work, we pursued different research directions with the aim of further improving the autonomous navigation capabilities supporting our UGVs operations.

The developed multi-robot framework is distributed and is based on a two level coordination strategy. Both topological and metric coordination allow to minimize interference and conflicts, which crucially affect UGVs activity. Continuous monitoring and replanning make the robots promptly react to environment changes and spatial conflicts. Operations in 3D are supported by a laser-based SLAM system.

The different capabilities of the developed multi-robot system allows to attain *persistence* in all the different phases of a typical long-term mission. Specifically, first, an autonomous *exploration* guides the robot team in co-operatively building a map of an unknown environment. Then, the built 3D map is saved and can be re-used by the end-users to interactively build and save a patrolling graph, i.e. a topological map representing a set of point of interests connected by safe paths. Next, *patrolling* can be started over the saved patrolling graph in order to make the robots monitor the assigned points of interest. Additionally, robots can be called to perform a *coverage* task in the explored environment in order to update the saved 3D map.

In particular, with respect to Year 3, the multi-robot patrolling strategy has been improved by *(i)* introducing prioritization of waypoints, *(ii)* revising the next goal selection strategies, *(iii)* making coordination protocols more robust, *(iv)* adding more control functionalities in the GUI and *(v)* successfully performing real-world patrolling experiments with three UGVs. We tested the patrolling system in different environments over the different TRADR exercises. A joint design analysis was conducted with end-users in order to identify the most suitable performance measure for patrolling tasks (average graph idleness minimization).

The new exploration and coverage capabilities were designed by casting the two-level coordination approach presented in 2.1 in the context of exploration. In the developed framework, exploration and coverage are based on the same receding horizon next-best-view approach. Here, the next view is selected over a local traversable region by maximizing the information gain over prospective trajectories, with a motion-predictive-like approach. During a mission, locations with higher priorities can be online assigned in order to bias the exploration (or coverage). Also for the exploration task, we discussed with end-users about the general mission requirements and possible design options.

In order to improve the autonomous navigation capabilities supporting the multi-robot system, we worked along the following directions of research and development (at the intersection of WP1, WP2 and WP4). First, we revised the onboard path planner pipeline. In particular, we analyzed and tested different point cloud segmentation and clustering methods in order to improve the underlying traversability analysis module and the recognition of stairs and ramps in the environment. Second, we integrated the adaptive traversal algorithm into the path planner, with the aim of pushing the autonomous navigation capabilities of our UGVs towards more challenging and harsh terrains. Third, we worked on a communication-aware path planning strategy (as documented in DR.2.4), which can be applied both to a single robot and to a multi-robot team. Fourth, we developed an RGBD-SLAM method, PLVS (Points, Lines, Volumetric mapping and incremental Segmentation), which can be used to robustly build a denser and coloured point cloud map and to enable a more advanced and accurate analysis of the terrain and surrounding objects.

In all the above developments, the V-REP simulator played a crucial role. Indeed, this was constantly improved in order to allow testing and validation of the entire multi-robot framework (path planning, patrolling, exploration and coverage), before real deployments.

The multi-robot framework has been fully integrated in the TRADR system. The underlying architecture is distributed over `nimbro_network`, a robust network transport layer for multi-master ROS systems. A flexible interface of the multi-robot framework with the main TRADR Orchestra system was developed in order to allow a fast deployment of the UGVs, even with a minimal laptop (cfr. DR.6.4).

In all the described WP4 work, the synergy with TRADR partners was an important ingredient. The research work on multi-robot localization and mapping, along with the development of the map saving functionality, has been obtained through a collaboration with ETHZ. The collaboration with KTH resulted in the development of the communication-aware path planning strategy (cfr. DR.2.4). Together with CTU, we integrated the adaptive traversal functionality in the path planner in order to autonomously drive UGVs over more challenging terrains. Last but not least, Fraunhofer continuously supported us in the integration of the functional architecture over `nimbro` and in the TRADR Core.

Implementing persistence in collaborative planning

The work performed in Year 4 by WP4 contributes to the overall objective of the TRADR project by providing a fully integrated and *ready-to-use* framework which allows a team of UGVs to perform exploration, patrolling and coverage tasks. A suitable GUI allows the end-user to (*i*) monitor the

tasks during their execution, *(ii)* pause and restart the tasks, *(iii)* build and save a patrolling graph, *(iv)* load and save a 3D map *(v)* define an exploration fence to contain robot exploration within a limited region of interest.

WP4 also provided the consortium with a version of the above framework which allows any user to run the TRADR system, under `nimbrowork`, in virtual simulation, as a tool for testing and benchmarking (see Subsection 1.3.6, Section 1.3).

Persistence

WP4 addressed persistence in Year 4 by developing different communication protocols under the `nimbrowork` network infrastructure to save/load onto/from the TRADR Core *(i)* the 3D maps of the environments created by the UGVs during exploration sorties, *(ii)* the trajectories followed by the robots during these exploration phases and *(iii)* the patrolling graph and *(iv)* a RSS map for communication-aware path planning.

In particular, a 3D point cloud map can be built and saved on one robot during a first exploratory sortie. A function has been developed in Orchestra (cfr. DR6.4) in order to distribute the saved map all over the active UGVs and on the TRADR core. Next, this map can be loaded, displayed on the main OCU and used to build and save a patrolling graph. All the relevant data structures are saved in the TRADR Core and render available for subsequent sorties.

The saved robot trajectories are used to refine the estimation of the point cloud normals in the map for both traversability assessment and path planning. Additionally, the saved trajectories can be used to build an estimate of the topology of the areas to be patrolled.

In this work, ETHZ developed the server side of the map saving/loading functionality. ROMA developed both the client side of the map saving/loading service and the service for robot trajectory saving. Fraunhofer contributed to the integration of the above work with `nimbrowork` and the TRADR Core (cfr. DR.6.4).

Contribution to the TRADR scenarios and prototypes

The research work of WP4 contributed to Year 4 Use-cases as documented on the related wiki pages on Redmine:

- (https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Multi-Robot_Use_Cases_Definition) (Annex 2.8)

- (https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Review_Yr3_Recommendations)
(Annex 2.7)

Annex 2.8 collects useful information for the identification of interesting multi-robot tasks within the TRADR project. The objective of this document is two-fold: (1) to clearly define a set of doable tasks which will be actually implemented for the reviews and demos and (2) to foster collaboration and discussions within the whole TRADR team.

We used Annex 2.7 in order to specifically interact with end-users about use-cases, to define together multi-robot strategies and discuss about design options, as required by Reviewers during Year 3 Review (see Section 1.2).

1 Tasks, objectives, results

1.1 Planned work

The planned work of WP4, in Year 4, concerning “Implementing persistence in collaborative planning” is described in Task T4.4. Task T4.4 achieves the objectives described in Milestone MS4.4. An excerpt of the description of both Task T4.4 and Milestone MS4.4, from the DoW of the project, is given below

Task T4.4: *The Goal of Task 4.4 is to model and implement persistence in multi-robot collaboration. Persistence covers all Tasks, and in Task 4.4 it becomes operative in the sense that it filters the cumulative data collected in successive sorties to maintain only the relevant information, and to transform it further into part of the individual and common knowledge.*

Milestone MS4.4: *This last milestone achieves the WP4 main objective, namely persistent collaboration across different sorties specifying a mission. MS4.4 proves that: (1) the collaborative planning framework, with its internal representation, its learning methods, its communication structure, can manage a small robot team to operate on a time horizon that goes beyond the single experiment; (2) that the knowledge of each robot is used for the success of the mission; (3) that communication is effectively operated by the team. It thus, shows, that collaboration can scale not in quantity but in endurance.*

1.2 Addressing reviewers’ comments

This section reports how in Year4 WP4 addressed the comments of the reviews.

Overall comment 1: *The patrolling solution presented in Dr4.3 for the TRADR system is not adequately linked to the requirements defined by end users regarding patrolling and, therefore, is not adequately justified. The consortium should clarify this design option in Yr4 deliverables of WP4 to better link requirements with the implemented solution.*

Answer to overall comment 1: We organized multiple joint discussions together with end-users focusing on the points highlighted by Reviewers. We clarified and identified together the main goals and desired performance measures of patrolling, exploration and coverage tasks. Requirements were discussed along with possible design options and implementation details. A dedicate page was prepared in the consortium wiki in order to support the discussion (https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Review_Yr3_Recommendations) (cfr. Annex 2.7). In the end, the interactions with end-users were very constructive. The current patrolling strategy and its GUI design

was discussed and approved by the end-users, as testified in the evaluations reported in Annex 2.7. The discussion also focused on the new exploration framework, which has been developed in this last Year 4. In this process, new suggestions were collected, such as the prioritization of points of interest.

Overall comment 2: *The integration in the TRADR system of the multi-robot collaboration framework presented in Yr2 review meeting still has been postponed to Yr4.*

Answer to overall comment 2: In Year 4, we developed and integrated in the TRADR architecture a new framework which allows a team of robots to perform patrolling, exploration and coverage tasks. We think this multi-robot system has completely superseded the framework presented in Yr2 (which was mainly used to explore different approaches and algorithms for coverage tasks).

1.3 Actual work performed

This section describes the research and development work of WP4 in Year 4 of the TRADR project. It is organized as follows. First, Subsection 1.3.1 introduces our framework for 3D multi-robot path planning and patrolling. Then, Subsection 1.3.2 describes how we extended that framework for achieving 3D multi-robot exploration. Next, Subsection 1.3.3 presents how we integrated the adaptive traversal function and the path planner. The work on point cloud segmentation and dense RGBD reconstruction is described in Subsections 1.3.4 and 1.3.5, respectively.

Finally, Subsection 1.3.6 presents software documentation history, installation instructions and user manuals of the various developed software packages. To this aim, dedicated wiki pages have been maintained and updated. These are available to all the project partners (cfr. Annexes 2.9 and 2.10).

1.3.1 3D Multi-robot Patrolling

This subsection presents our framework for 3D multi-robot path planning and patrolling. This was already introduced in Year 3 (cfr. Annex 2.1). During Year 4, we improved this framework by *(i)* revising our work from a theoretical perspective¹ *(ii)* introducing prioritization of waypoints, *(iii)* revising the next goal selection strategies, *(iv)* making coordination protocols more robust, *(v)* adding more control functionalities in the GUI and *(vi)* successfully performing real-world patrolling experiments with three UGVs over the TRADR architecture. We also tested and validated the patrolling

¹We considered the suggestions from the last TRADR Review. Moreover, we submitted our work to the AURO Journal and received other precious comments from the Journal Reviewers.

system in different environments over the different TRADR exercises (Year3 Review, TJEx and TEval).

The code developed for this framework will be released as open source, as reported in Sect. 1.3.6.

The subsection is organized as follows. First, Paragraph 1.3.1.1 introduces the patrolling model. Then, Paragraphs 1.3.1.2–1.3.1.6 shortly present its main components. Next, Paragraph 1.3.1.7 and 1.3.1.8 introduces the concepts of topological/metric conflicts and the two-level coordination strategy. The functional architecture is described in Paragraph 1.3.1.9. Paragraph 1.3.1.10 presents the two different procedures for patrolling graph building. Afterwards, Paragraph 1.3.1.11 briefly introduces the 3D GUI designed for end-user interaction purposes. Finally, patrolling results are cross-reference in Paragraph 1.3.1.12. More details can be found in Annex 2.1.

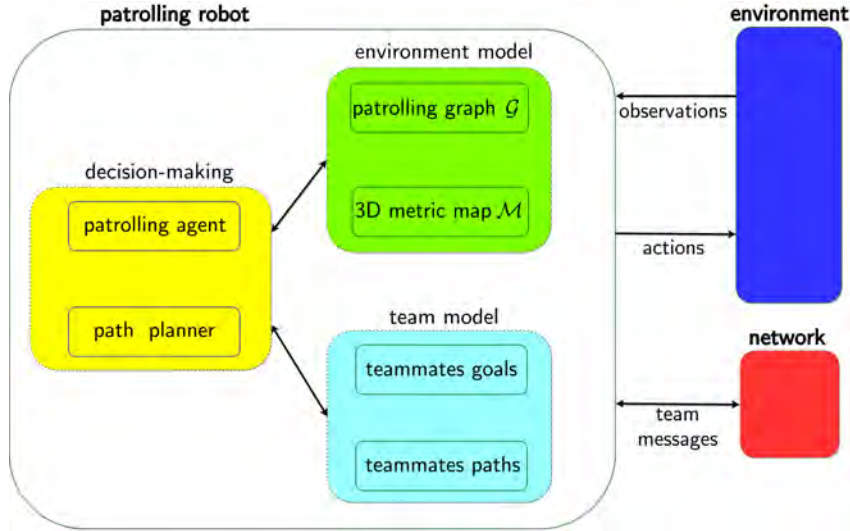


Figure 1: Patrolling robot model.

1.3.1.1 Patrolling Model

A team of $m \geq 2$ ground patrolling robots is called to patrol a 3D environment. A set of locations of interest is assigned and must be continuously visited in order to monitor their surroundings. The team objective is to maximize the visit frequency of each assigned location.

The main components of a *patrolling robot* are represented in Fig. 1. A patrolling robot interacts with its environment through observations and actions, where an observation consists of a vector of sensor measurements and an action corresponds to a robot actuator command. Team messages are exchanged with teammates over a network for sharing knowledge and

decisions in order to attain team collaboration.

Decision making is achieved by the patrolling agent and the path planner, basing on the available information stored in the environment model and the team model. In particular, the *environment model* consists of a topological map \mathcal{G} , aka patrolling graph, and a 3D metric map \mathcal{M} . The *team model* represents the robot belief about the current plans of teammates (goals and planned paths).

The 3D *environment* \mathcal{W} is a compact connected region of \mathbb{R}^3 . The robots move on a 3D terrain, which is identified as a compact and connected manifold \mathcal{S} in \mathcal{W} . We denote by $T = [t_0, \infty) \subset \mathbb{R}$ a *time interval*, where $t_0 \in \mathbb{R}$ is the starting time. More details about the robots configuration space are provided in 2.1.

The main components of the patrolling robots are introduced in the following subsections.

1.3.1.2 Patrolling Graph and Patrolling Agent

A *patrolling graph* \mathcal{G} is a topological graph-like representation of the environment to be patrolled. Namely, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is an undirected connected graph, with \mathcal{N} a set of nodes and $\mathcal{E} \subseteq \mathcal{N}^2$ a set of edges.

A node $n_i \in \mathcal{N}$ is associated to a 3D region of interest $\mathcal{R}(n_i) \subset \mathcal{W}$ and to a *priority weight* $w(n_i) \in \mathbb{R}^+$. In particular, $\mathcal{R}(n_i)$ is a ball of pre-fixed radius $R_v \in \mathbb{R}$ centred at the corresponding position $\mathbf{p}(n_i) \in \mathcal{S}$.

An edge $e_{ij} \in \mathcal{E}$ between node n_i and n_j denotes the existence of a safe path τ_{ij} connecting the regions $\mathcal{R}(n_i)$ and $\mathcal{R}(n_j)$. The length of such a path is used as edge *travel cost* $c(e_{ij}) \in \mathbb{R}^+$.

A node $n_j \in \mathcal{N}$ is *visited* at time $t \in T$ if a robot centre lies inside the associated region $\mathcal{R}(n_j)$ at t .

The *instantaneous idleness* $I_j(t) \in \mathbb{R}^+$ of a node $n_j \in \mathcal{N}$ at time $t \in T$ is $I_j(t) = w(n_j)(t - t_l)$ where t_l is the most recent time in $[t_0, t]$ the node was visited by a robot. The priority $w(n_j) \in \mathbb{R}^+$ is used to locally “dilate” or “contract” time at node n_j . We assume $I_j(t_0) = 0$ for each node n_j in \mathcal{G} .

Considering the idleness $I_j(t)$ of a node n_j in a time subinterval $[t_1, t_2] \subset T$, we compute its average idleness $I_j^a[t_1, t_2] = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} I_j(t) dt$, its standard deviation $I_j^\sigma[t_1, t_2] = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} (I_j(t) - I_j^a[t_1, t_2])^2 dt$ and its maximum value $I_j^M[t_1, t_2] = \arg\max_{t \in [t_1, t_2]} I_j(t)$.

The *average graph idleness* of \mathcal{G} is

$$I_{\mathcal{G}}[t_0, t] = \frac{1}{N} \sum_{j=1}^N I_j^a[t_0, t] \quad (1)$$

where $N = |\mathcal{N}|$ is the total number of nodes in \mathcal{G} .

Table 1: Table of patrolling broadcast messages.

Broadcast message	Description	Affected data in receiving robot h
$\langle j, \text{reached}, n \rangle$	robot j has reached goal node n	node n idleness is zeroed; the j -th tuple in team model $\mathcal{T}^{(h)}$ is reset
$\langle j, \text{visited}, n \rangle$	robot j is visiting node n	node n idleness is zeroed
$\langle j, \text{planned}, n \rangle$	robot j has planned node n as goal	j -th tuple in team model $\mathcal{T}^{(h)}$ is reset
$\langle j, \text{selected}, n, \tau, c \rangle$	robot j has selected node n as goal, τ is the planned path to n and c the corresponding path length	the j -th tuple in team model $\mathcal{T}^{(h)}$ is filled with (n, τ, c)
$\langle j, \text{aborted}, n \rangle$	robot j aborted goal node n	the j -th tuple in team model $\mathcal{T}^{(h)}$ is reset
$\langle j, \text{idleness}, \mathcal{I}^{(j)}(t) \rangle$	robot j shares its current idleness estimations $\mathcal{I}^{(j)}(t) = \langle I_1^{(j)}(t), \dots, I_N^{(j)}(t) \rangle$	the current idleness estimations $\mathcal{I}^{(h)}(t)$ are synchronized with $\mathcal{I}^{(j)}(t)$

The *patrolling plan* π of a robot is defined as an infinite sequence $\{(n_k, t_k)\}_{k=0}^{\infty}$, where $n_k \in \mathcal{N}$ denotes the k -th node visited at time $t_k \in T$ by the robot. A *team patrolling strategy* $\Pi = \{\pi_1, \dots, \pi_m\}$ collects the patrolling plans of all the robots in the team.

Patrolling objective. In our framework, the goal of the robot team is to cooperatively plan a team patrolling strategy that minimizes the average graph idleness $I_G[t_0, t]$ at all times $t \in T$.

An instance of the patrolling agent runs on each robot h and is responsible of cooperatively generating the patrolling plan π_h according to the above patrolling objective. A pseudo-code description of the patrolling agent is provided in Annex 2.1.

As reported in Annex 2.7, we discussed the above patrolling objective with our end-users.

1.3.1.3 Metric Map and Path-Planning

Each robot of the team is equipped with a 3D laser range-finder and is able to localize in a global map frame, which is shared with its teammates.

In our framework, each robot uses a 3D point cloud as a metric representation \mathcal{M} of the environment. A *multi-robot traversability cost* function $\text{trav} : \mathbb{R}^3 \rightarrow \mathbb{R}$ is defined on each point of \mathcal{M} . This is used to associate a navigation cost $J(\tau)$ to each safe path τ .

Given the current robot position $\mathbf{p}_r \in \mathbb{R}^3$ and a goal position $\mathbf{p}_g \in \mathcal{S}$, the *path planner* computes the safe path τ^* which minimizes the navigation cost $J(\tau)$ and connect \mathbf{p}_r with \mathbf{p}_g . The path planner reports a failure if a safe path connecting \mathbf{p}_r with \mathbf{p}_g is not found.

See Annex 2.1 for further details.

1.3.1.4 Shared Knowledge Representation

Each robot of the team stores and updates its individual representation of the world state. Robots are able to *broadcast messages* in order to share knowledge, decisions and achievements with teammates. Some of these messages can be lost during transmissions. Table 1 summarizes the used broadcast messages along with the conveyed information/data (see Annex 2.1 for further details). The general broadcast message format is $\langle robot_id, message_type, data \rangle$.

At $t_0 \in T$, a robot loads as input the 3D map \mathcal{M} and the patrolling graph \mathcal{G} , then, it internally maintains an instance of these representations. In particular, we denote by $\mathcal{M}^{(h)}$ and $\mathcal{G}^{(h)}$ the local instances of \mathcal{M} and \mathcal{G} in robot h , respectively.

Since the environment is dynamic, robot h updates its individual 3D map $\mathcal{M}^{(h)}$ by using the last acquired 3D laser data. This allows the path planner to safely take into account new environment changes.

At the same time, robot h updates its patrolling graph $\mathcal{G}^{(h)}$ by using the received broadcast messages and the path planner output. Specifically, the travel cost $c^{(h)}(e_{ij})$ of an edge e_{ij} in $\mathcal{G}^{(h)}$ is locally updated when a new path is computed between the two corresponding nodes n_i and n_j .

Additionally, robot h locally maintains an idleness estimate $I_j^{(h)}(t)$ for each node n_j in $\mathcal{G}^{(h)}$. We denote by $\mathcal{I}^{(h)}(t) = \langle I_1^{(h)}(t), \dots, I_N^{(h)}(t) \rangle$ the vector of estimated idlenesses in robot h . Every time a robot visits/reaches a node n_j , a *visit/reach* message is broadcast and each receiving robot h correspondingly updates its local idleness estimate $I_j^{(h)}(t)$. Clearly, since broadcast messages may be lost, the idleness estimates $I_j^{(h)}(t)$ may not correspond to the actual idleness values. In order to mitigate this problem, each robot continuously broadcasts an *idleness* message at a fixed frequency $1/T_I$. Such messages are used to synchronize the idleness estimations amongst robots (see Annex 2.1 for further details).

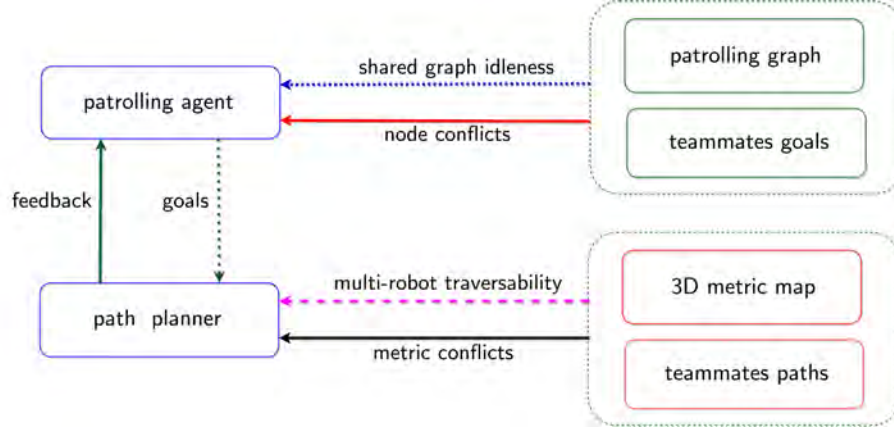
The above information sharing mechanism implements a *shared idleness* representation which allows team cooperation by minimizing useless actions such as re-visiting nodes just inspected by teammates.

1.3.1.5 Team Model

In order to cooperate with its team and manage conflicts, robot h maintains an internal belief representation of teammate plans (aka *team model*) by using a dedicated table

$$\mathcal{T}^{(h)} = \langle (n_g^1, \tau^1, c^1, t^1), \dots, (n_g^m, \tau^m, c^m, t^m) \rangle \quad (2)$$

which stores for each robot j : its selected goal node $n_g^j \in \mathcal{N}$, the last computed safe path τ^j to n_g^j , the corresponding travel cost $c^j \in \mathbb{R}^+$ (i.e. the

Figure 2: The two-level *patrolling* strategy implemented on each robot.

length of τ^j) and the timestamp $t^j \in T$ of the last message used to update (n_g^j, τ^j, c^j) . The table $\mathcal{T}^{(h)}$ is updated by using all the received broadcast messages. Old invalid message data are cleaned off from $\mathcal{T}^{(h)}$ by assigning a pre-fixed expiration time to each received message. See Annex 2.1 for further details.

1.3.1.6 System Architecture

The patrolling plan π of a robot can be pre-computed *offline*, i.e. before starting the patrolling execution, or *online*, i.e. by planning and visiting a new node at each patrolling step k .

In a *centralized* system, the team patrolling strategy $\{\pi_1, \dots, \pi_m\}$ is computed by a central control robot (i.e. the *leader*) and communicated to all its teammates. Conversely, in a *decentralized* system, a central leader does not exist. Different levels of decentralization are possible [109] and spans from hierarchical to distributed architectures. In a *distributed* system, each robot computes its own patrolling plan by exchanging information and coordination messages with teammates (see Sect. 1.3.1.4).

Our patrolling system is online and distributed. In particular, an instance of the *patrolling agent* algorithm runs on each robot and is responsible of online generating its patrolling plan. Namely, at each patrolling step k , the patrolling agent plans a new *goal node* n_k in \mathcal{G} . In this process, a robot exchanges messages with its teammates in order to attain *coordination* (avoid conflicts) and *cooperation* (avoid inefficient actions).

1.3.1.7 Topological and Metric conflicts in Patrolling

In this paragraph, we introduce the definitions of topological and metrical conflicts. These support the proposed two-level coordination strategy.

A *topological conflict* between two robots is defined on the patrolling graph \mathcal{G} . This occurs when two patrolling agents select the same node $n_i \in \mathcal{G}$ as goal (*node conflict*) or plan to traverse the same edge $e_{ij} \in \mathcal{G}$ (*edge conflict*).

On the other hand, metric conflicts are defined in the 3D Euclidean space where two robots are in *interference* if their centres are closer than a pre-fixed *safety distance* D_s . It must hold $D_s \geq 2R_b$, where R_b is the bounding radius of each robot, i.e. the radius of its minimal bounding sphere. A *metric conflict* occurs between two robots if they are in interference or if their planned paths may bring them in interference².

1.3.1.8 Two-Level Coordination Strategy for Patrolling

Our patrolling strategy is distributed and supported on two levels: topological and metric.

The patrolling agent acts on the *topological* strategy level by selecting the next goal node n_g on \mathcal{G} . *Cooperation* is attained by using the shared idleness representation in this selection process (see Sect. 1.3.1.4).

The path planner acts on the *metric* strategy level (see Figure 2) by computing the best safe path from the current robot position to $\mathbf{p}(n_g)$ by using its internal 3D map $\mathcal{M}^{(h)}$.

The patrolling agent guarantees *topological coordination* by continuously monitoring and negotiating possibly incoming node conflicts. In case a group of robots select the same goal (node conflict), the robot with the smaller travel cost (i.e. path length) actually goes, while the other robots stop and re-plans a new node.

The path planner guarantees *metric coordination* by applying a multi-robot traversability function. This induces a prioritized path planning, in which robots negotiate metric conflicts by preventing parts of their planned paths from intersecting.

The continuous interaction between the patrolling agent and the path planner plays a crucial role. When moving towards the position $\mathbf{p}(n_g)$, the path planner continuously re-plans the best traversable path till the robot reaches the goal. During this process, if a safe path is not found, the path planner stops the robot, informs the patrolling agent of a *path planning failure* and the patrolling agent re-plans a new node. On the other hand, every time the path planner computes a new safe path, its length is used as travel cost by the patrolling agent to resolve possible node conflicts.

In our view, the two-way strategy approach allows (i) to simplify the topologically based decision-making, (ii) to reduce interferences and manage possible deadlocks. In fact, while the patrolling agent focuses on the

²That is, the distance between the closest pair of points of the two planned paths is smaller than D_s .

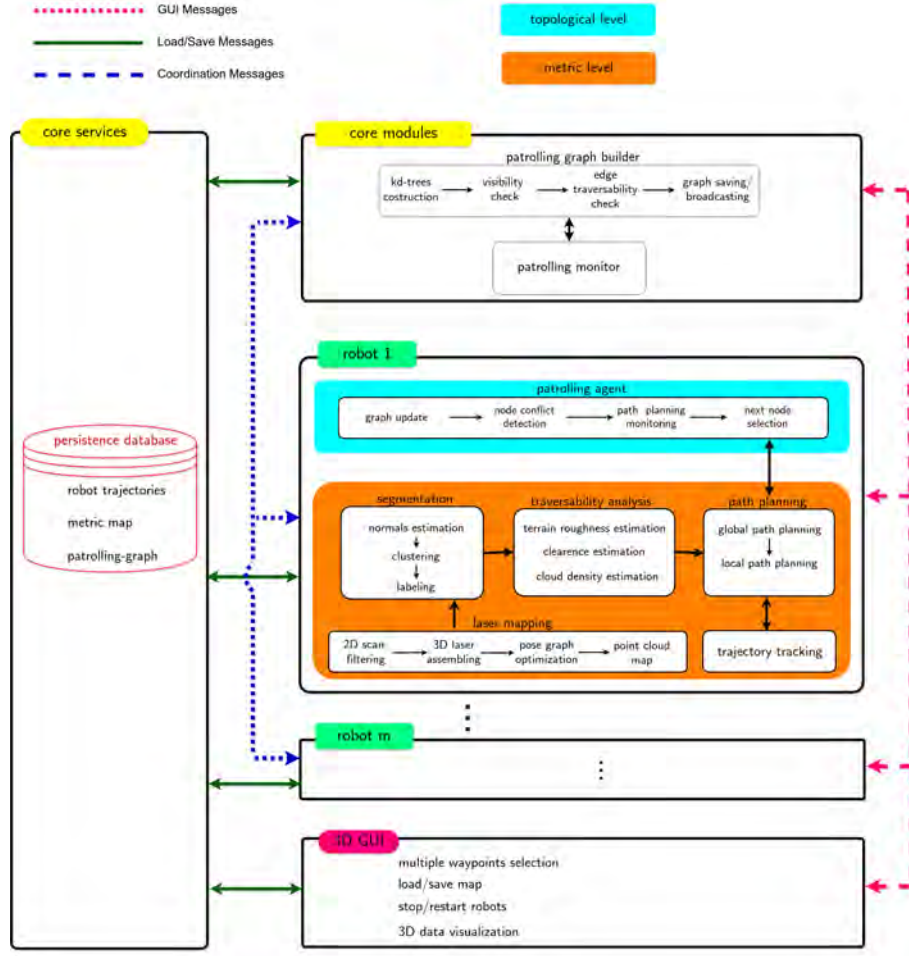


Figure 3: The functional diagram of the multi-robot *patrolling* system. Robots share the same internal software architecture. The legend on the top left represent the different kind of exchanged messages.

most important graph aspects (shared idleness minimization and node conflicts resolution), the path planner takes care of possible incoming metric conflicts due to unmanaged topological edge conflicts. Moreover, where the path planner strategy may fail alone in arbitrating challenging conflicts, the patrolling agent intervenes and reassigns tasks in order to better redistribute robots over the graph. These combined strategies minimize interferences by explicitly controlling node conflicts and planning on multi-robot traversability map.

More details about the patrolling strategy are provided in 2.1.

1.3.1.9 Patrolling Software Design

A functional diagram of the multi-robot patrolling system is reported in Fig. 3. The main blocks are listed below.

- The *robots*, each one with its own ID $\in \{1, \dots, m\}$: these have the same internal architecture and host the on-board functionalities which concern decision-making and data processing aspects both at topological level and at metric level. According to Paragraph 1.3.1.4, each robot maintains and updates an instance of the patrolling graph and of the metric map in its internal memory.
- The *core services*, hosted in the main central computer: these manage the multi-robot system persistence database and allow specific modules to load/save map, trajectories and patrolling graphs from/into the central database (for re-using relevant data along different missions).
- The *core modules*, also hosted in the central computer: these include the patrolling graph builder and the patrolling monitor. The first can build a patrolling graph from a user assigned sequence of waypoints or from a saved history of robot trajectories. The built patrolling graph is then distributed to all the robots and saved in the central database. The patrolling monitor continuously checks the current status of the patrolling activities and records relevant data for monitoring and benchmarking.
- The *multi-robot 3D GUI* hosted on one OCU (Operator Control Unit): this is based on RVIZ and allows the user *(i)* to select multiple waypoints, which can be fed to the path planners or to the patrolling graph builder *(ii)* to visualize relevant point cloud data, maps and robot models *(iii)* to stop/restart robots when needed *(iv)* to trigger the loading/saving of maps and robot trajectories.

As shown in Fig. 3, the various modules in the architecture exchange different kind of messages. These are grouped in the following types.

- Coordination messages: these are mainly exchanged amongst robots in order to achieve coordination and cooperation. For convenience, the patrol monitor records an history of these messages.
- GUI messages: these are exchanged with the 3D GUI and include both control messages and visualization data.
- Load/save messages: these are exchanged with the core services and contain both loaded and saved data.

1.3.1.10 The Patrolling Graph Construction

The patrolling graph can be built *(1)* manually by the user or *(2)* automatically by processing robot trajectories collected during past sorties.

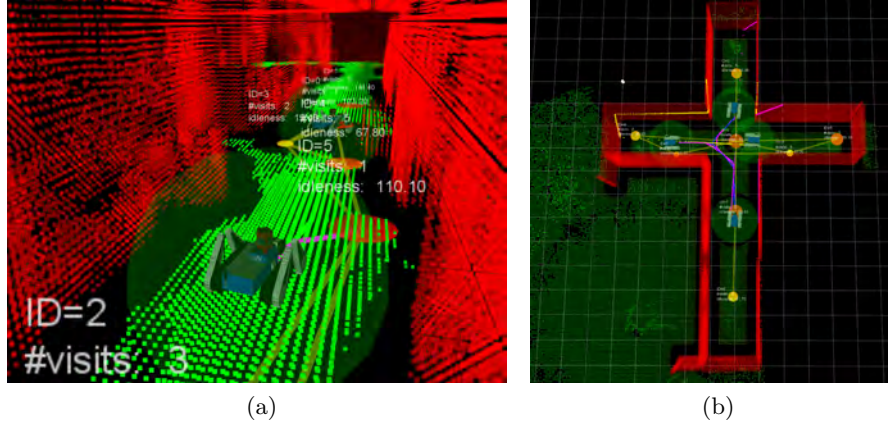


Figure 4: RVIZ Visualization (see Paragraph 1.3.1.11) of a team of TRADR UGVs (4a) negotiating paths along a corridor and (4b) coordinating their planning strategy in a critical case. Point clouds are colored on the basis of the approach for traversability assessment; yellow spots connected to straight lines of the same color denote nodes of the patrolling graph

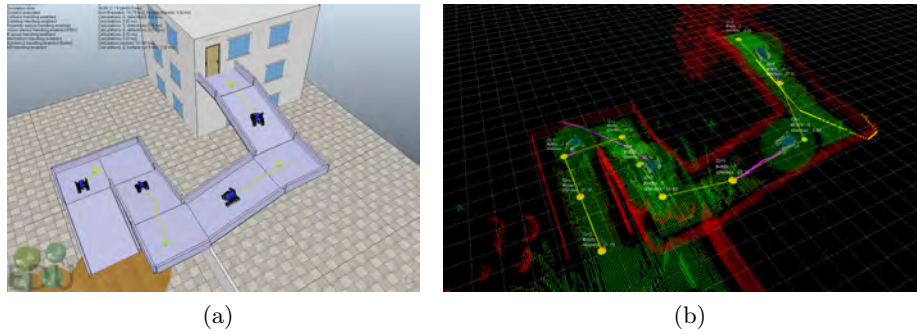


Figure 5: (5a) A team of TRADR UGVs deployed in virtual simulation (V-REP [32]) over nimbro and the TRADR Core; (5b) RVIZ Visualization (see Paragraph 1.3.1.11) of scenario in Figure 5a.

In the first mode, the user is provided with a 3D GUI. This GUI allows to load a saved map and to interactively “draw” a patrolling graph on the corresponding traversable map.

In the second mode, an application automatically estimates a potential patrolling graph by condensing and downsampling past robot trajectories with an approach similar to [72].

1.3.1.11 3D Interface: RVIZ data visualization and commands

The 3D GUI allows to visualize the distinct robot maps and the distinct robot modules in the same main `/map` frame³. This is possible since, in each robot, we added an intermediate `/robot_name/map` frame between the main `/map` frame and the `/odom` frame and we remapped each robot link frame by adding `/robot_name` prefix.

1.3.1.12 Patrolling Results

The results we obtained with our patrolling framework are presented in the work 2.1. In particular, some videos of V-REP simulations and real-world experiments can be found at <https://sites.google.com/a/dis.uniroma1.it/3d-cc-patrolling/>. Notably, during Year 4 we succeeded to perform new experiments with a system of three UGVs, integrated in the nimbro [92] architecture and managed by using a minimal Orchestra system (see DR.6.4).

Our main conclusions are the following. The presented two-level coordination strategy is general and can be used as a solid base to develop new strategies which are able to (i) optimize a defined performance measure on the patrolling graph and (ii) avoid deadlocks by explicitly managing the occurrence of interference and space conflicts.

Our multi-robot patrolling algorithm is fully integrated with a 3D SLAM algorithm, traversability analysis and coordinated path planning. This allows UGVs operations on a 3D uneven terrain.

We demonstrate competitive performance without deadlocks in both simulation and real world experiments, enabling 4 robots to simultaneously operate in realistic simulations and 3 robots in real world experiments.

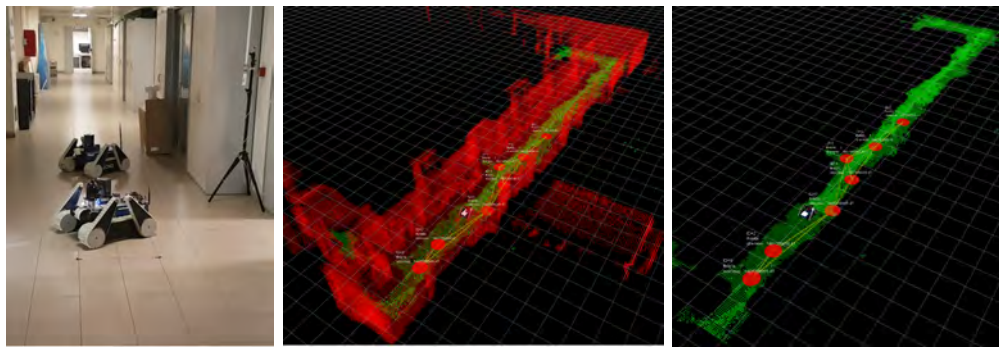
1.3.2 3D Multi-robot Exploration

This subsection presents our multi-robot system for exploration and coverage. This was designed by casting the two-level coordination strategy of our patrolling system (see Subsection 1.3.1) in the context of 3D exploration. The resulting distributed technique succeeds to minimize and explicitly manage the occurrence of conflicts and interferences in the exploration team. Each robot selects where to scan next by using a receding horizon

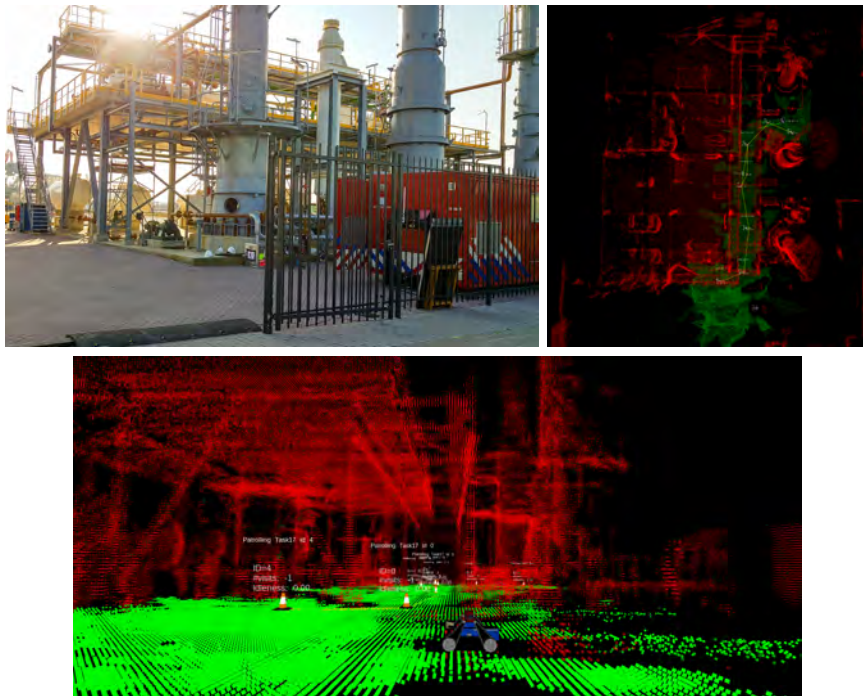
³Here, we use the ROS topics naming convention.



(a) DIAG ramp



(b) DIAG corridor



(c) RDM Deltalinqs Training plant

Figure 6: Experimented scenarios with real UGVs.

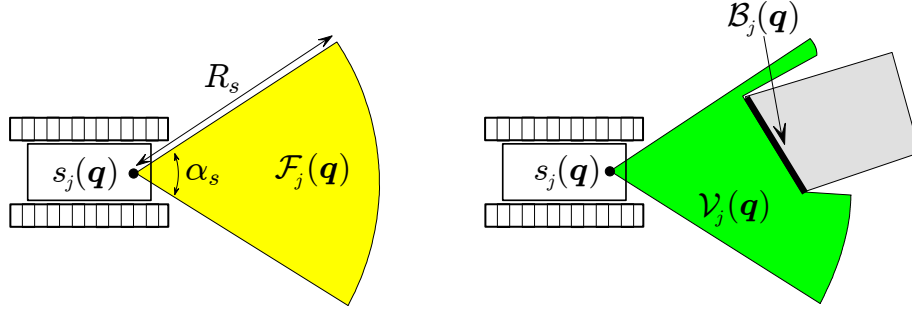


Figure 7: *Left*: robot j with its sensor center $s_j(\mathbf{q})$ and the associated field of view $\mathcal{F}_j(\mathbf{q})$ at configuration \mathbf{q} . *Right*: the view $\mathcal{V}_j(\mathbf{q})$ and the visible obstacle boundary $\mathcal{B}_j(\mathbf{q})$. The view $\mathcal{V}_j(\mathbf{q})$ is the set of “free” points visible from $s_j(\mathbf{q})$ and lying in $\mathcal{F}_j(\mathbf{q})$. The obstacle boundary $\mathcal{B}_j(\mathbf{q})$ is the set of “obstacle” points visible from $s_j(\mathbf{q})$ and lying in $\mathcal{F}_j(\mathbf{q})$.

next-best-view approach [13]. Here, a sampling-based tree is directly expanded on segmented traversable regions of the terrain 3D map. Locations with higher priorities can be online assigned in order to bias the exploration process. The presented framework can be also used to perform coverage tasks in the case a 3D map of the environment is a priori provided as input.

The code developed for this framework will be released as open source, as reported in Sect. 1.3.6.

1.3.2.1 Exploration Task

A team of $m \geq 2$ ground exploration robots wake up in an unknown environment and are constrained to move on an uneven terrain. The team has to perform an exploration, i.e. cooperatively cover the largest possible part of the environment with sensory perceptions [39]. The output of an exploration process is a 3D model of the environment.

The 3D *environment* \mathcal{W} is a compact connected region of \mathbb{R}^3 . The robots move on a 3D terrain, which is identified as a compact and connected manifold \mathcal{S} in \mathcal{W} .

A detailed definition of the sensor model and the exploration task are given in Annex 2.2. Here, we sketch the basic concepts. The sensor model of the generic robot j is shown in Fig. 7. In particular, the sensor field of view $\mathcal{F}_j(\mathbf{q})$ is a circular sector with apex $s_j(\mathbf{q})$, opening angle α_s and radius R_s , where the latter is the *perception range*.

Robot j plans a sequence of view configurations $\mathbf{q}_j^1, \mathbf{q}_j^2, \dots, \mathbf{q}_j^k$, so that at step $k \geq 1$, its *explored region* is

$$\mathcal{E}_j^k = \mathcal{E}_j^{k-1} \cup \mathcal{V}_j(\mathbf{q}_j^k).$$

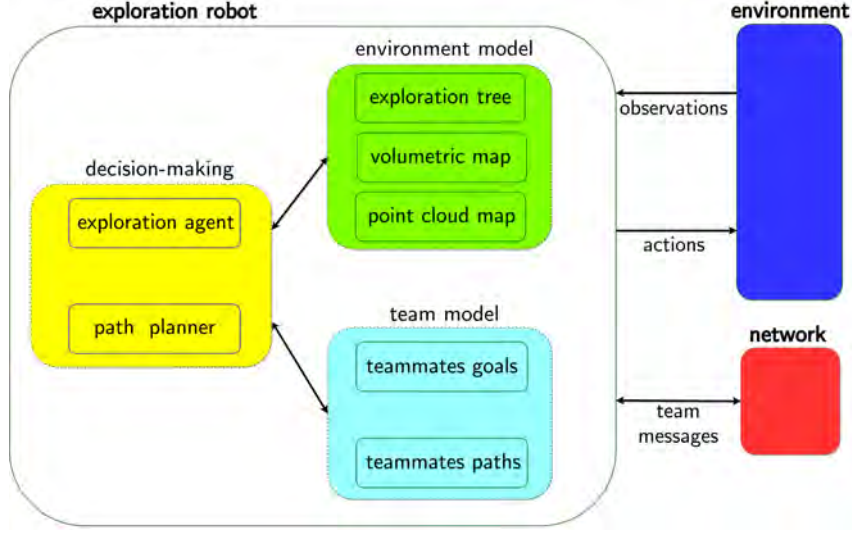


Figure 8: The exploration robot model.

Robot j starts from an endogenous knowledge $\mathcal{E}_j^0 \subset \mathbb{R}^3$ at the initial configuration \mathbf{q}_j^0 . If we consider the whole robot team, the overall explored region at step k is

$$\mathcal{E}^k = \bigcup_{j=1}^m \mathcal{E}_j^k = \bigcup_{j=1}^m \mathcal{E}_j^0 \cup \left(\bigcup_{j=1}^m \bigcup_{i=0}^k \mathcal{V}_j(\mathbf{q}_j^i) \right). \quad (3)$$

An *exploration plan* π_j of robot j is a finite sequence of view configuration $\pi_j = \{\mathbf{q}_j^k\}_{k=0}^{l_j}$, where l_j is the length of π_j . A *team exploration strategy* $\Pi = \{\pi_1, \dots, \pi_m\}$ collects the exploration plans of the robots in the team. Here, $l = \max(l_1, \dots, l_m)$ is the length of the strategy Π .

Exploration objective. The robots must cooperatively plan an exploration strategy Π of minimum length l such that \mathcal{E}^l is maximized. In particular, \mathcal{E}^l is *maximized* at step l if there is no robot j that can plan a new “safe and reachable” view configuration \mathbf{q} such that $\mathcal{E}^l \subset (\mathcal{E}^l \cup \mathcal{V}_j(\mathbf{q}))$.

Other factors, such as the resulting map “accuracy” could be taken into account in the exploration strategy. We further develop this in Annex 2.2.

1.3.2.2 Exploration Robot Model

Fig. 8 presents the main components of an *exploration robot*. This presents many similarities to the patrolling robot of Fig. 1. Decision making is achieved by the exploration agent and the path planner, basing on the available information stored in the environment model and the team model. In particular, the *environment model* consists of a topological map, aka exploration tree, and two different metric maps: the volumetric map and the point

Environment Models in Robot j	Description
Exploration tree \mathcal{K}_j	Topological map of the environment: it stores the history of view configurations $\{\mathbf{q}_k\}$ in the form of a tree.
Volumetric map \mathcal{H}_j	Representation of the explored region \mathcal{E}^k : it is used to compute the information gains of candidate view configurations.
Point cloud map \mathcal{M}_j	Representation of the detected surfaces of the environment: it is used by the path planner to compute safe paths over segmented traversable regions and by the exploration agent to generate candidate view configurations.

Table 2: Table of environment models in robot j .

cloud map. As in Paragraph 1.3.1.1, the *team model* represents the robot belief about the current plans of teammates (goals and planned paths).

The main components of the exploration robots are introduced in the following subsections. Table 2 reports the used symbols along with their short descriptions.

1.3.2.3 Exploration Tree and Exploration Agent

During the exploration process, an *exploration tree* \mathcal{K}_j is built by robot j . A node of \mathcal{K}_j is referred to as *view node* and represents a view configuration. An edge between two view nodes corresponds to a safe path joining them. \mathcal{K}_j is rooted at \mathbf{q}_j^0 . At each forwarding step, a new node corresponding to \mathbf{q}^{k+1} and a new edge representing a path from \mathbf{q}^k to \mathbf{q}^{k+1} are added in \mathcal{K}_j .

We consider an exploration tree as a topological map representation of the explored environment. Each node of the tree represents an explored local region contained within a ball of radius R_s (the sensor perception range) centred at the associated view configuration.

The *exploration agent* is responsible of online generating an exploration plan according to the defined exploration objective (cfr. Par. 1.3.2.1).

1.3.2.4 Point Cloud Mapping and Path-Planning

The exploration and patrolling systems adopt the same kind of point cloud map for representing environment surfaces. Moreover, they use the same path planner and multi-robot traversability function (see Paragraph 1.3.1.3). In particular, for exploration tasks, the 3D point cloud map \mathcal{M}_j of robot j is not loaded as input at starting time (as it is used in the patrolling system) but it is incrementally built as a metric representation of the detected

environment surfaces $\bigcup_{j=1}^m \bigcup_{i=0}^k \mathcal{B}_j(\mathbf{q}_j^i)$ (recall Fig. 7).

Table 3: Table of exploration broadcast messages.

Broadcast message	Content description	Affected data in receiving robot h
$\langle j, \text{reached}, \mathbf{p} \rangle$	robot j has reached goal position \mathbf{p}	j -th tuple in team model \mathcal{T}_h is reset
$\langle j, \text{planned}, \mathbf{p} \rangle$	robot j has planned node \mathbf{p} as goal	j -th tuple in team model \mathcal{T}_h is reset
$\langle j, \text{selected}, \mathbf{p}, \boldsymbol{\tau}, c \rangle$	robot j has selected node \mathbf{p} as goal, $\boldsymbol{\tau}$ is the planned path and c the corresponding path length	j -th tuple in team model \mathcal{T}_h is filled with $(\mathbf{p}, \boldsymbol{\tau}, c)$
$\langle j, \text{aborted}, \mathbf{p} \rangle$	robot j has aborted goal node \mathbf{p}	j -th tuple in team model \mathcal{T}_h is reset
$\langle j, \text{scan}, \mathbf{v}, \mathbf{q} \rangle$	robot j has acquired new 3D scan data \mathbf{v} at \mathbf{q}	3D scan data is integrated in volumetric map \mathcal{H}_h
$\langle j, \text{tree}, \mathcal{K}_j \rangle$	robot j shares its exploration tree \mathcal{K}_j	\mathcal{K}_h is compared with \mathcal{K}_j to check if a map synchronization with robot j is needed

1.3.2.5 Volumetric Map and Information Gain

A volumetric map \mathcal{H}_j is incrementally built by robot j to represent the explored region \mathcal{E}^k and associate an information gain to each safe configuration. Specifically, \mathcal{H}_j is a probabilistic occupancy gridmap, stored in the form of an Octomap [50]. This partitions the environment in *free*, *occupied* and *unknown* cells with a pre-fixed resolution. Such a model allows to nicely model an environment populated by low-dynamic objects [104].

We compute the information gain $I(\mathbf{q}, k)$ as the volume of the unknown cells of \mathcal{H}_j which are visible at step k from the robot at \mathbf{q} , following the approaches in [67, 45].

1.3.2.6 Shared Knowledge Representation and Update

Each robot of the team stores and updates its individual representation of the world state.

In particular, robot j incrementally builds its individual point cloud map \mathcal{M}_j by using the acquired scans (see 2.1). This allows the path planner to safely take into account explored terrain extensions and possible environment changes.

At the same time, robot j updates its volumetric map \mathcal{H}_j by integrating its acquired 3D scans and the *scan* messages received from teammates. As mentioned above, some of the scan messages can be lost and \mathcal{H}_j may only partially represent the actual explored region \mathcal{E}^k .

We are currently working on a procedure to mitigate this problem: each robot continuously broadcasts a *tree* message at a fixed frequency $1/T_M$. In particular, robot h uses a *tree* message of robot j in order to estimate if the map \mathcal{H}_h sufficiently overlaps with \mathcal{H}_j or it missed the integration of some *scan* message data. A pseudocode description of this procedure is reported in Annex 2.2 this verifies if the Euclidean projections of \mathcal{K}_j and \mathcal{K}_h sufficiently overlap each other. If a sufficient overlap is not verified, a

synchronization and merge procedure⁴ is triggered between the maps of the two robots i and j relying on the approaches described in [28, 29].

The above information sharing mechanism allows to implement a *shared information gain*, i.e. each robot computes the information gain on the basis of a distributed shared knowledge. This enforces team cooperation by minimizing useless actions such as exploring regions already visited by teammates.

1.3.2.7 Team Model

The team model of an exploration robot is very similar to the one used by a patrolling robot (see Paragraph 1.3.1.5). Specifically, the *planning state of the team* (aka team model) is represented by the following table

$$\mathcal{T}_j = \langle (\mathbf{p}_1, \boldsymbol{\tau}_1, c_1, t_1), \dots, (\mathbf{p}_m, \boldsymbol{\tau}_m, c_m, t_m) \rangle. \quad (4)$$

This stores for each robot h its selected goal position $\mathbf{p}_h \in \mathcal{S}$ (instead of storing a node as in patrolling), the last computed safe path $\boldsymbol{\tau}_h$ to \mathbf{p}_h , the associated travel cost $c_h \in \mathbb{R}^+$ (i.e. the length of $\boldsymbol{\tau}_h$) and the timestamp t_h of the last message used to update $(\mathbf{p}_h, \boldsymbol{\tau}_h, c_h)$. The table \mathcal{T}_j is updated by using all the received broadcast messages. Old invalid message data are cleaned off from \mathcal{T}_j by assigning a pre-fixed expiration time to each received message. See the work 2.2 for further details.

1.3.2.8 Topological and Metric conflicts in Exploration

A *topological conflict* between two robots i and j is defined with respect to their exploration trees \mathcal{K}_i and \mathcal{K}_j . Specifically, a *node conflict* occurs when robot i and j try to add a new view node in the same spatial region at the same time, respectively n_i in \mathcal{K}_i and n_j in \mathcal{K}_j . We identify this situation when the corresponding positions $\mathbf{p}(n_i) \in \mathbb{R}^3$ and $\mathbf{p}(n_j) \in \mathbb{R}^3$ are closer than a certain distance $D_g \leq R_s$ (the sensor perception range).

For exploration and patrolling tasks, metric conflicts are defined in the same way (see Paragraph 1.3.1.7).

1.3.2.9 Two-Level Coordination Strategy for Exploration

Our exploration strategy maintains the same two-level approach introduced in Paragraph 1.3.1.8 (see Figure 9).

The exploration agent acts on the *topological* strategy level by suitably planning and adding a new view node n_g (corresponding to a view configuration) on its exploration tree. *Cooperation* is attained by using the shared information gain (see Sect. 1.3.2.6) in this planning process.

⁴This is a work in progress.

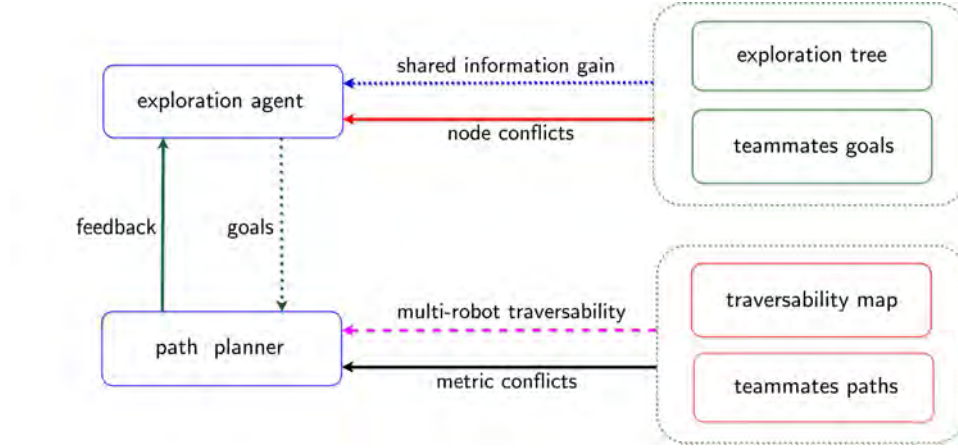


Figure 9: The two-level *exploration* strategy implemented on each robot.

The path planner acts on the *metric* strategy level (see Figure 9) by computing the best safe path from the current robot position to the goal position $\mathbf{p}(n_g)$ by using its individual point cloud map (see Annex 2.1 for further details).

The exploration agent guarantees *topological coordination* by continuously monitoring and negotiating possibly incoming node conflicts (see Annex 2.2). In case two or more robots plan new view configurations close to each other (node conflict), the robot with the smaller path length actually goes, while the other robots stop and re-plans a new view node.

As explained in Paragraph 1.3.1.8, the path planner guarantees *metric coordination*. Also in this case, the continuous interaction between the exploration agent and the path planner plays a crucial role.

More details about the exploration strategy are provided in Annex 2.2.

1.3.2.10 Coverage Task

Coverage and exploration tasks have the same objective: robots have to cover the environment with sensory perceptions. But while coverage robots have a full prior knowledge of the environment \mathcal{W} (which can be used to plan safe motions), exploration robots must online discover \mathcal{W} and restrict their motions within the known terrain.

Our team of exploration robots can be used to perform a coverage task. To this aim, we provide them with a point cloud map \mathcal{M} representing the full environment. Specifically, at t_0 , robot j sets $\mathcal{M}_j = \mathcal{M}$ and $\mathcal{H}_j = \emptyset$. During the coverage task, the volumetric map \mathcal{H}_j is used as in exploration: it is incrementally built in order (i) to represent the regions of the environment covered so far and (ii) to compute the information gain of candidate view configurations. The map \mathcal{M}_j is used by the path planner to plan safe paths.

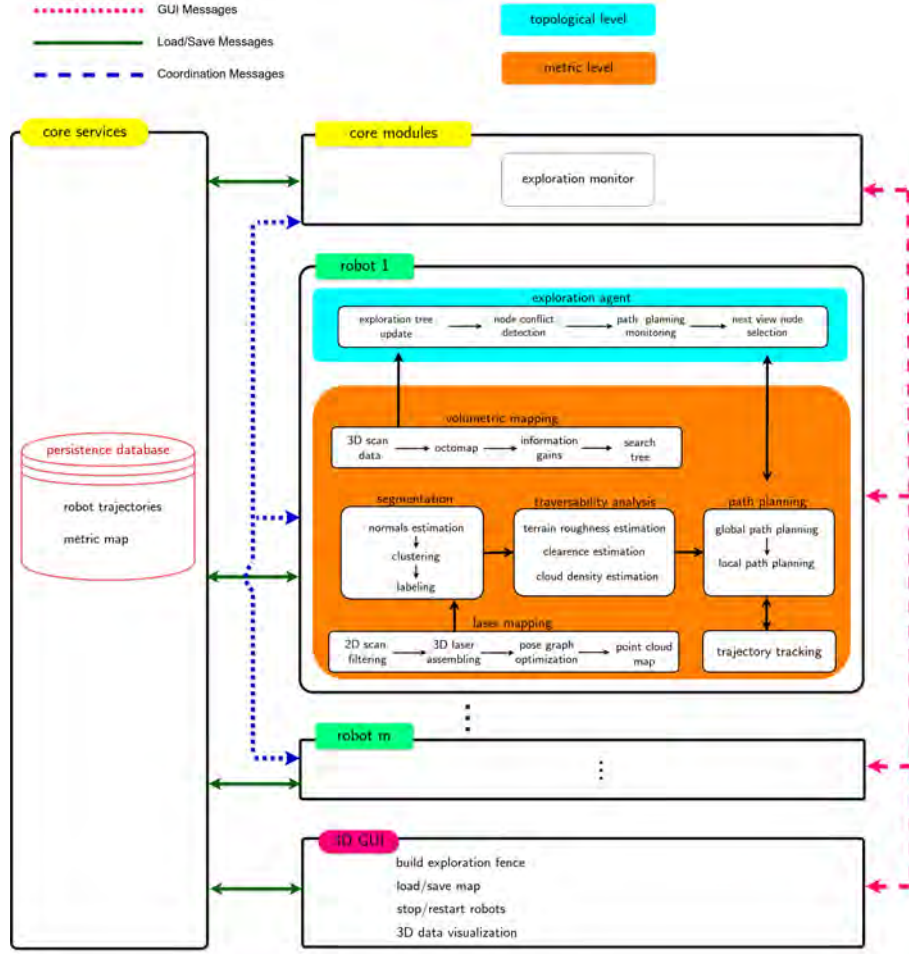


Figure 10: The functional diagram of the multi-robot *exploration* system. Robots share the same internal software architecture. The legend on the top left represent the different kind of exchanged messages.

Clearly, such an approach does not take advantage of the full prior knowledge of the environment, which in principle allows to pre-compute an optimal team strategy. Nonetheless, our framework can be conveniently used when the environment undergoes low-dynamic changes which can invalidate a precomputed plan. In fact, our 3D mapping is capable of continuously integrating detected changes and our online decision-making correspondingly adapts robot behaviour to the new environment model.

1.3.2.11 Exploration Software Design

A functional diagram of the multi-robot exploration system is reported in Fig. 10. This architecture have many blocks in common with the patrolling

system architecture shown in Fig. 3.

- The *robots*, each one with its own ID $\in \{1, \dots, m\}$: these have the same internal architecture and host the on-board functionalities which concern decision-making and data processing aspects both at topological level and at metric level. According to Paragraph 1.3.2.6, each robot maintains and updates an environment model in its internal memory. With respect to the architecture in Fig. 3, the exploration robots host a volumetric mapping module and an exploration agent (instead of the patrolling agent).
- The *core services*, hosted in the main central computer: these represent the multi-robot system persistence. These allow specific modules to load and save map and robot trajectories (along with other TRADR data structures). With respect to the architecture in Fig. 3, for sake of clarity, we removed the patrolling graph from the central database (which is here useless).
- The *core modules*, also hosted in the central computer: these includes the exploration monitor (instead of the patrolling monitor). This continuously checks the current status of the exploration activities and records relevant data for monitoring and benchmarking.
- The *multi-robot 3D GUI* hosted on one OCU (Operator Control Unit): this is based on RVIZ and allows the user *(i)* to build an *exploration fence*, which consists of a set of points defining a polygonal bounding region limiting the explorable area *(ii)* to visualize relevant point cloud data, maps and robot models *(iii)* to stop/restart robots when needed *(iv)* to trigger the loading/saving of maps and robot trajectories in the central memory. We built this RVIZ interface starting from the patrolling RVIZ GUI by adding the new needed controls for exploration tasks.

1.3.2.12 Exploration Results

Our exploration framework was tested during the last TRADR evaluation exercise in Rotterdam, in November 2017. The full TRADR system was evaluated by the GB firefighters end-users in the RDM Delinquents Training plant (see Fig. 11a). In particular, Fig. 11b shows the TRADR OCU displaying the GUI with a map and robot camera feedback.

We were able to run different exploration experiments and build a map of the plant with two UGVs. Fig. 12 shows two maps obtained at two different stages of an exploration process. During those runs the two UGVs started at the same location. In order to attain a multi-robot localization, a background process was designed to globally align the two robot maps in 3D and, in case of success, provide corrections to the mutual position estimates. This procedure provided good results when the pose graphs of

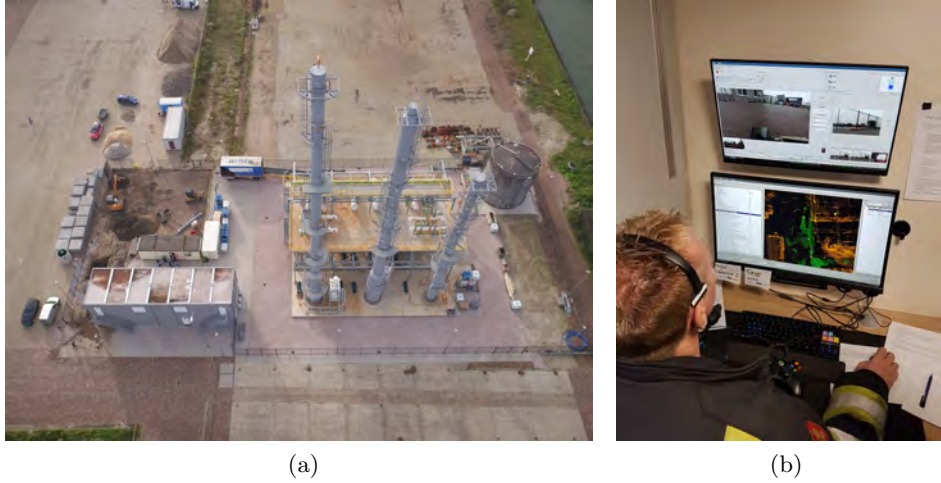


Figure 11: Left: the RDM Deltalinqs Training plant used for TEVAL exercise in November 2017. Right: a fireman operating the GUI interface for monitoring path planning, patrolling and exploration.

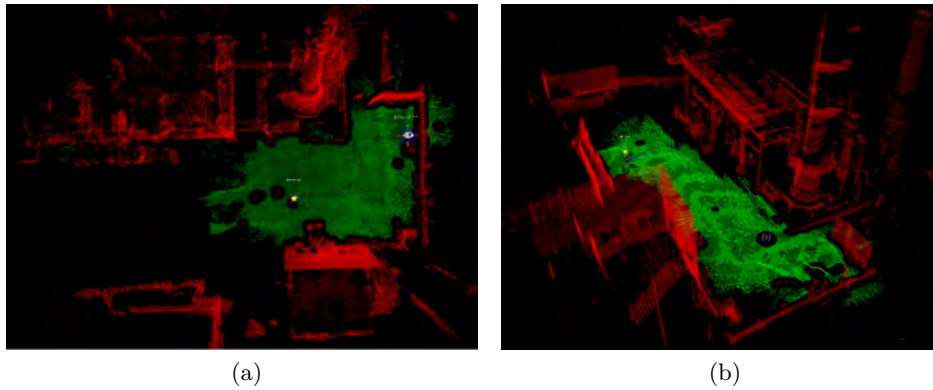


Figure 12: An experiment with a team of two UGVs in the RDM Deltalinqs Training plant. Two maps obtained in two different phases of a collaborative exploration.

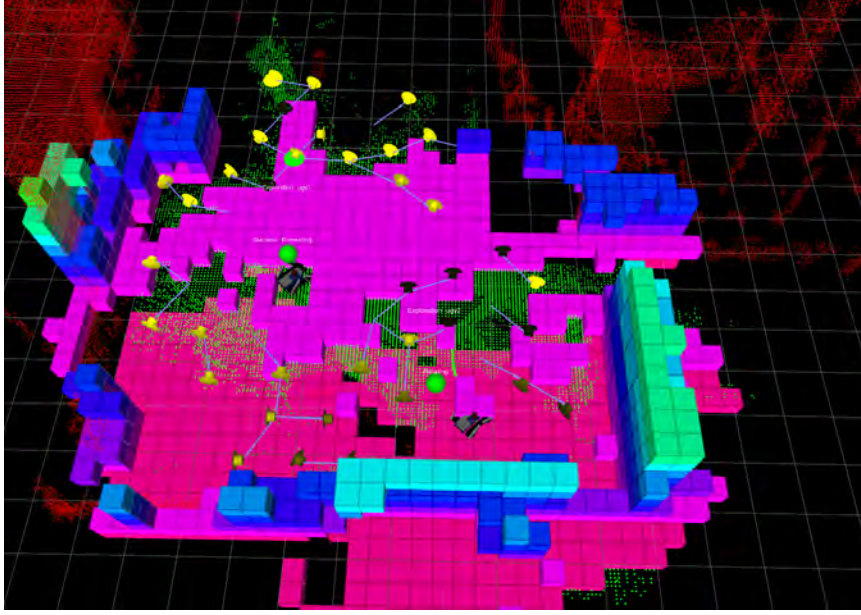


Figure 13: An experiment at the RDM Deltalinqs Training plant. A team of two UGVs was used during the mission. In this figure, point cloud maps and volumetric maps are overlaid. The cubes represent the obstacle cells of the volumetric map \mathcal{H}_j built by one of the robots. The green points represent segmented *traversable regions* of the point cloud map \mathcal{M}_j . The red points represent segmented *obstacles* on the point cloud map \mathcal{M}_j . The two search trees of the robots are shown (see Annex 2.2 for further details): these collect candidate view configurations, represented as small yellow arrows in the figure. In particular the lighter the yellow of the arrows, the higher the associated utility value.

the two registered maps were mutually consistent (up to a roto-translation) and did not present strong distributed deformations.

We observed that map visualization is the most demanding networking functionality of the TRADR system. This is only required on the 3D GUI of the central core if a user want to monitor exploration activities (see Fig. 10). In this context, the NIMBRO network transport layer was crucial for achieving almost smooth map data transfers.

Fig. 14 shows a simulation run with a team of two robots. An extensive simulation analysis is currently in progress in order to assess and analyse in detail the performances of the presented exploration method.

More results and details about multi-robot exploration running in V-REP simulation can be found at the following link:

https://drive.google.com/open?id=1X65BoVY7L5Ve4GhC1nJ3t_Iw18Uhrq0x.

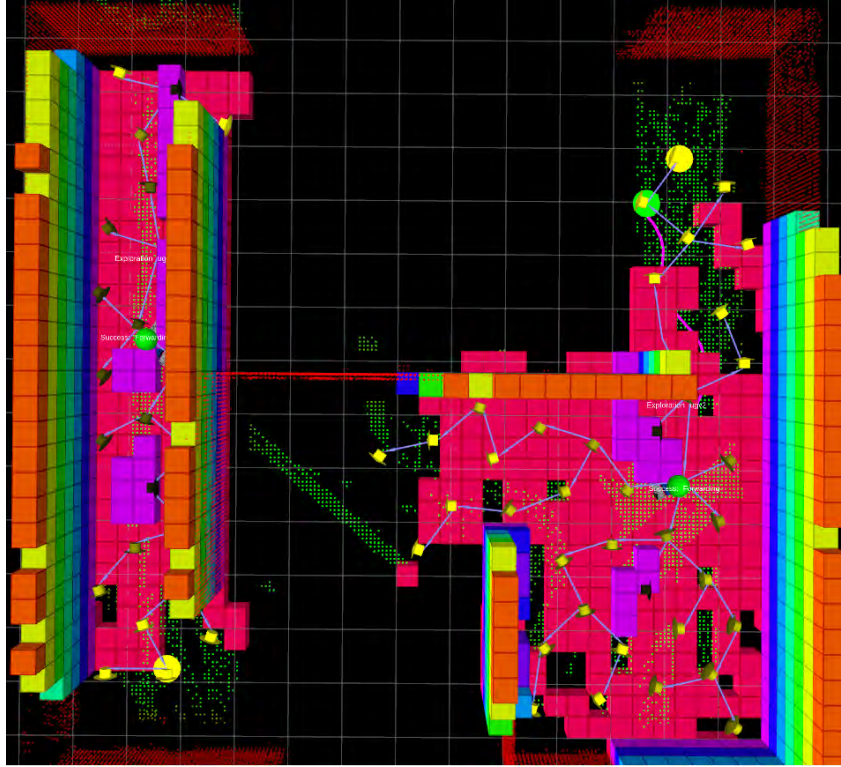


Figure 14: A V-REP simulation with a team of two exploration robots. The two different volumetric maps of the robots are overlapped and shown in the picture. The robots start in different locations, knowing their initial mutual positions. As for the volumetric map, only the obstacle cells are shown. The search tree of each robot is also shown. Each search tree is expanded only of the traversable portion of the point cloud map.

1.3.3 Integration of Adaptive Traversal Function in Path Planning

CTU and ROMA joined their efforts in integrating the Adaptive Traversal (AT) algorithm into the path planner. The goal of this integration is to push the autonomous navigation capabilities of our UGVs towards more challenging and harsh terrains.

Before the integration, the path planner did not control the flippers leaving them in the default position all the time. This may restrict the reachable area of the UGV but it also simplifies collision checking. As a result, the (local) path planner is able to continuously re-plan a new safe path at a very fast pace and the UGV readily reacts to possible dynamic changes in the environment.

We integrated the AT algorithm on the local level of the path planner

by using a decoupled approach with the aim of maintaining computational efficiency. In this regard, our sampling-based local path planner defers to the AT algorithm the selection of the most appropriate flipper configuration for each sampled configuration.

In particular, as reported in 2.1, for implementation and efficiency reasons we make use of a global and a local path planners. Given a set of 3D waypoints as input, the *global* path planner is in charge of *a)* checking the existence of a traversable path joining them and *b)* minimizing a *traversability cost function* along the computed path. This cost function combines together the multi-robot traversability cost along with an optional task dependent cost function. Once a global path solution τ_g is found, the *local* path planner *continuously* re-plans a traversable path τ_l that safely drives the robot from its current configuration \mathbf{q} to the first configuration of τ_g intersecting a sphere of radius R_l centred at \mathbf{q} .

The local path planner and the AT algorithm work together to compute a local safe path τ_l by using a decoupled approach. Let $\mathbf{q} = [\mathbf{q}_b, \mathbf{q}_f]^T \in \mathcal{C}$ denote a full configuration of the UGV, where $\mathbf{q}_b \in \mathcal{C}^b$ is the configuration of the baselink in $SE(3)$ and $\mathbf{q}_f \in \mathcal{C}^f$ denotes the flipper configurations (i.e. a set of four flipper angles). A sampling-based tree (randomized A*) is expanded in the full configuration space \mathcal{C} . A kinematic model of the full UGV body (baselink+flippers) and the available 3D map of the environment are used for collision checking during the process. At each step of the sampling-based tree expansion:

1. first, the local path planner generates a new sample configuration \mathbf{q}_b which places the baselink over the traversable points of the map;
2. then, the AT algorithm chooses the most appropriate configuration \mathbf{q}_f , which best adapt the flippers to the 3D model of the terrain;
3. next, collision checking is performed in order to validate the full generated configuration \mathbf{q} .

Indeed, this decoupled approach allows to efficiently compute a solution path in the full configuration space \mathcal{C} by actually using its projection \mathcal{C}^b as the actual search space (where new sample configurations are generated). Additionally, this approach inherits the advantages of the AT algorithm both at planning time and execution time.

In fact, at execution time, the AT algorithm is consistently called to work in parallel with the trajectory tracking algorithm, with the same decoupled approach. Specifically, once a path is planned and must be executed, the trajectory tracking controls the UGV baselink along the projected planned path in \mathcal{C}^b while the AT correspondingly re-computes the actual flipper configurations by using laser data.

In conclusion, the basic technical difficulties of the integration have already been overcome. In order to attain a robust UGV navigation in all

the conditions (especially over very harsh terrains), there is still the need of some work in bug-fixing and fine-tuning of the parameters.

1.3.4 Point Cloud Segmentation and Clustering for Traversability Analysis

Point cloud segmentation and clustering are crucial ingredients in terrain traversability analysis for UGV navigation. In Annex 2.1, we presented a path planning pipeline in which, as a basis, map points are segmented in order to estimate a traversability of the terrain. First, geometric features such as surface normals and principal curvatures are computed and organized in histogram distributions. Then, clustering is applied on 3D coordinates of points, mean surface curvatures and normal directions [72, 37]. As a result, a classification (*labeling*) of the 3D map in regions such as *walls*, *terrain*, *surmountable obstacles* and *stairs/ramps* is obtained.

In Annex 2.6, we analysed, developed and tested different segmentation methods and point clustering algorithms with the following main goals:

- Identify in the literature methods which could be efficiently implemented and run on-board of our UGVs with their limited computational resources;
- Implement and test these methods on our laser datasets. It is worth noting that the maps generated by our UGVs present in some cases a significant noise, which can make segmentation a challenging task.
- Compare the performances of our traversability analysis algorithms with the new implemented methods.
- Find direction of improvements for our traversability analysis module.

In particular, we focused on methods for segmenting planes, stairs and ramps in a 3D point cloud map. These methods have an important role in many industrial scenarios in which our UGVs are supposed to operate.

Given the limited computational capabilities of our UGVs, in this research and engineering work, we only considered efficient geometric-based algorithms. At present time, the literature offers a huge amount of data-driven methods. Most of them are based on deep learning, can only be implemented by using a powerful GPU (and a significant amount of dedicated memory) and are specifically designed for vision tasks (e.g. visual object classification). On our UGVs, we do not have advanced hardware resources and the input consists of a large amount of 3D laser data, which need to be fast processed.

In Annex 2.6, each of the examined algorithms was tested in combination with 3D edge extraction methods. Beyond classical segmentation and clustering algorithms (e.g. conditional Euclidean clustering, RANSAC with plane and line model fitting, super-voxel clustering) we also tested

approaches such as Difference-of-Normals segmentation [56], the Hough-transform-based plane segmentation described in [15] and the more recent constrained plane fitting segmentation method in [80].

We found that the segmentation methods in [15] and in [80] are very promising and succeed to segment planes well in a number of situations. Nonetheless, these methods depend on many parameters and are very sensitive to their changes. In fact, we verified that, in many cases, the smallest change of one parameter can invalid the whole map segmentation. Moreover, in order to attain the best results, these parameters should be manually tuned and adapted to the scenario at hand.

As an additional result, we found that the Difference-of-Normals method [56] works best for segmenting out stairs, as a specialized and robust edge extraction method. Nonetheless, this approach is computational intensive⁵ and cannot be flexibly used for recognizing also planes, ramps and in general traversable regions. This implies that it could only be used as an additional segmentation module on our UGVs if additional on-board computational resources were available.

This work brought the following conclusions:

- we experienced that the general usability of the analysed and tested methods is very limited with the available TRADR datasets⁶; these methods are very sensitive to parameters change; in order to have good results, these parameters should be tuned and adapted each time to the dataset/scenario at hand.
- For our purposes (in the TRADR scenarios), the segmentation and clustering algorithms used in our UGVs show on the average good results, with the best flexibility and efficiency-accuracy trade-off.

Clearly, further analyses with more complex algorithms (e.g. based on DNNs and CNNs) and the availability of a dedicated hardware with GPU on our UGVs would bring our research on different directions and results.

1.3.5 3D Reconstruction and Incremental Segmentation by using RGBD-SLAM

In this section, we present PLVS, a research project at the intersection of WP1, WP2 and WP4. PLVS stands for Points, Lines, Volumetric mapping, incremental Segmentation. The goal of the PLVS project is to create a novel and comprehensive RGB-D SLAM framework which can reliably run onboard of small-sized robotic platforms and is able to generate a volumetric map of the surrounding environment. An efficient incremental (geometric-based) segmentation is available as a basis for enabling advanced object detection and semantic mapping.

⁵It requires the computation of normals at two different resolutions.

⁶These were recorded during exercises or in scenarios of interest.

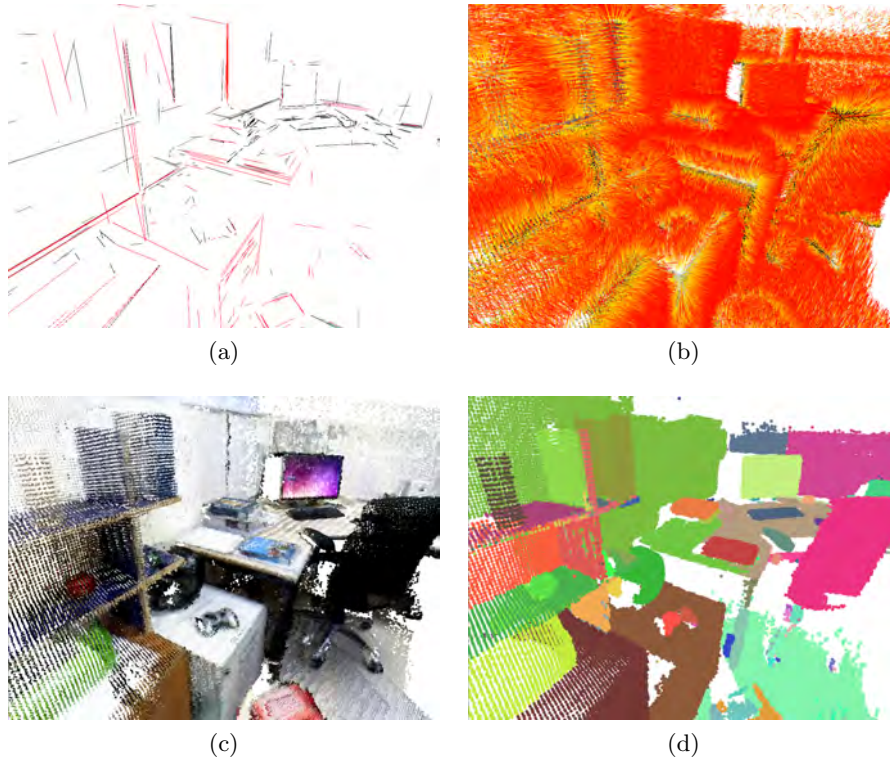


Figure 15: Details of a 3D reconstruction with PLVS (a) line segments (b) normals (c) point cloud (d) segmented point cloud.

We think PLVS can enable the construction of accurate and coloured maps which can better support the autonomous navigation of UGVs and enhance the end-user situational awareness. In particular, traversability analysis could benefit from the richer information offered by a denser and coloured map. This can clearly enable a more robust segmentation of stairs/ramps and planes (cfr. Sect. 1.3.4). Moreover, the mutual robot localization could be enhanced by the vision-based place recognition algorithm of PLVS (visual bag of words) to detect large loop closures. Additionally, a dense coloured 3D map can better represent the environment and be exploited by the end-users for more detailed assessments and decision-making.

In more detail, PLVS is a real-time system which leverages sparse RGB-D SLAM, volumetric mapping and 3D unsupervised incremental segmentation. The system can run entirely on CPU or can profit by available GPU computational resources in order to efficiently extract features from images. The SLAM module is keyframe-based and relies on the extraction and tracking of sparse points and segment lines as features. Volumetric mapping runs in parallel with respect to the SLAM front-end and allows to obtain a 3D reconstruction of the explored environment by fusing point clouds backprojected



Figure 16: The 3D reconstruction and the segmented map obtained in a big office environment by using PLVS.

from keyframes. Different volumetric mapping methods are supported and integrated in PLVS. A novel reprojection error is used for bundle-adjusting line segments. This error takes advantage of depth information in order to better stabilize the position estimate of line segment endpoints and can be also used with stereo camera systems. An unsupervised incremental segmentation method (based on point normals and on the information extracted from 3D lines) is implemented and integrated in the PLVS framework.

PLVS was designed as a modular framework, i.e. its different capabilities are organized in divisions which can be enabled/disabled and configured by users in different ways. This allows to finely trade-off map accuracy/resolution versus CPU load and to adapt the framework capabilities to the system at hand.

We achieved state of the art performance by running the system entirely on the CPU of a laptop with a typical Intel Core-i7. We obtained good results on an NVIDIA Jetson TX1 board by (i) using a CUDA implementation of the feature extraction module, (ii) decreasing the working resolution of the volumetric mapping and (iii) disabling some more CPU-demanding capabilities.

It is worth noting that current state-of-the-art methods such as Elastic-Fusion [106] use fully optimized GPU code and, in order to properly run, they usually require advanced computational capabilities which are only available on top-end NVIDIA GPUs. Clearly, such advanced GPU capabilities are not available on small-sized robotic platforms given their usual SWaP (Size, Weight and Power) limitations.

In table 4 and table 5, we present some preliminary performances of the PLVS system obtained on a laptop with Intel Core-i7 (see below). For each evaluation dataset, we report the median results over five runs in order to account for the non-deterministic nature of the multi-threading system.

In particular, in table 4, we report the obtained tracking accuracy on the TUM datasets [100]. We compare therein PLVS to the state-of-the-art

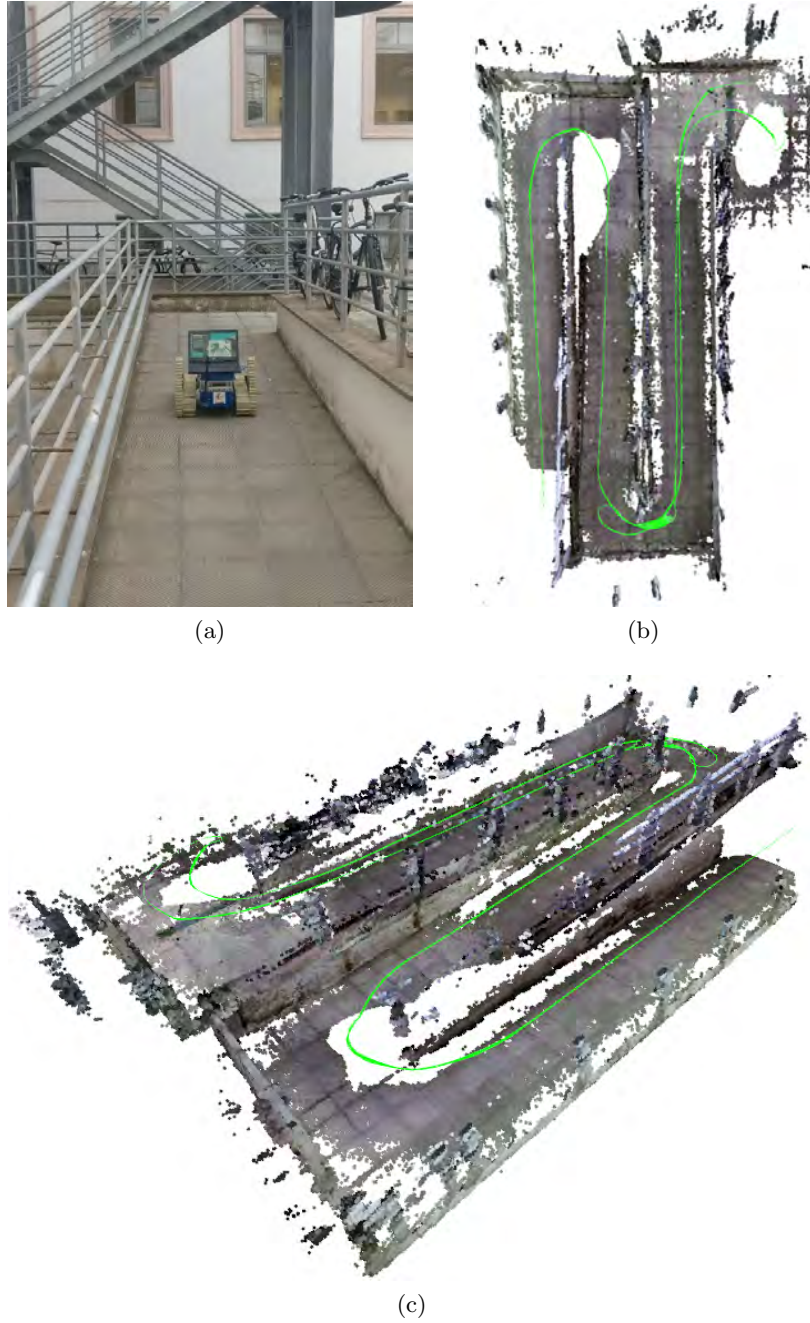


Figure 17: (a) The UGV equipped with a realsense R200, (b) and (c) the volumetric map built by PLVS in the ramp-experiment.

Dataset	PLVS-RMSE	ElasticFusion-RMSE	FrameDrops
freiburg1 360	0.095	0.108	1.6%
freiburg1 desk	0.015	0.02	3.1%
freiburg1 desk2	0.039	0.048	3.2%
freiburg1 floor	-	-	11.8%
freiburg1 plant	0.019	0.022	1.3%
freiburg1 room	0.058	0.068	0.7%
freiburg1 rpy	0.018	0.025	4.0%
freiburg1 teddy	0.043	0.083	1.3%
freiburg1 xyz	0.009	0.011	0.8%
freiburg2 360 hemisphere	0.144	-	2.2%
freiburg2 360 kidnap	-	-	1.2%
freiburg2 coke	0.511	-	2.0%
freiburg2 desk	0.013	0.071	2.5%
freiburg2 dishes	0.032	-	2.2%
freiburg2 large no loop	0.525	-	1.8%
freiburg2 large with loop	-	-	2.4%
freiburg2 metallic sphere	0.721	-	1.7%
freiburg2 metallic sphere2	0.121	-	1.2%
freiburg2 pioneer 360	-	-	61.6%
freiburg2 pioneer slam	-	-	52.7%
freiburg2 pioneer slam2	-	-	52.3%
freiburg2 pioneer slam3	-	-	32.3%
freiburg2 rpy	0.003	0.015	2.1%
freiburg2 xyz	0.006	0.011	1.5%
freiburg3 cabinet	-	-	4.1%
freiburg3 large cabinet	0.043	0.099	3.9%
freiburg3 long office household	0.011	0.017	4.9%
freiburg3 nostructure notexture far	-	-	4.7%
freiburg3 nostructure notexture near withloop	-	-	3.9%
freiburg3 nostructure texture far	0.031	0.074	4.0%
freiburg3 nostructure texture near withloop	0.022	0.016	3.5%
freiburg3 structure notexture far	0.021	0.03	3.5%
freiburg3 structure notexture near	0.02	0.021	3.7%
freiburg3 structure texture far	0.011	0.013	4.7%
freiburg3 structure texture near	0.015	0.015	4.9%
freiburg3 teddy	-	0.049	4.2%

Table 4: Absolute Tracking Error (ATE) RMSE obtained on the TUM datasets [100]. For comparison purposes, we report the performances of PLVS and of ElasticFusion [106]. As for PLVS, it is worth noting that we used the same constant set of parameters in all the environments. It is worth noting that, although the capturing device provides RGB-D frames at 30Hz, a number of sequences are missing a certain amount of frames that were simply never recorded: this value is represented in FrameDrops column. This percentage values are provided in [106].

System	kt0	kt1	kt2	kt3
PLVS-Octree	0.008	0.019	0.028	0.008
PLVS-Chisel	0.013	0.024	0.025	0.011
PLVS-FastFusion	0.078	0.123	0.105	0.093
PLVS-Octomap	0.107	0.047	0.025	0.012
PLVS-VoxelGrid	0.057	0.062	0.066	0.040
ElasticFusion	0.007	0.007	0.008	0.028
DVO-SLAM	0.032	0.061	0.119	0.053
RGB-DSLAM	0.044	0.032	0.031	0.167
MRSMap	0.061	0.140	0.098	0.248
Kintinuous	0.011	0.008	0.009	0.150

Table 5: A comparison of the surface reconstruction accuracies which were obtained on the ICL-NUIM datasets [46]. The reported values represent the average distance (in meters) of each reconstructed point to the nearest surface in the ground truth 3D model.

ElasticFusion [106]. It is worth mentioning that while ElasticFusion (and also other methods) adopts a different set of optimized parameters in each environment, we used the same constant set of parameters for PLVS in all the datasets.

In table 5, we report the obtained surface reconstruction accuracies on the ICL-NUIM datasets [46] and we compare PLVS with other state-of-the-art methods [59, 99, 106, 105, 35].

It is worth mentioning that on December 2017, we prepared a short paper presentation of PLVS. This was submitted to NVIDIA for an **NVIDIA GPU Grant Request**. The request was **approved** on December 30th and NVIDIA was happy to support our work with the donation of a Jetson TX2 board.

The code developed for this framework will be released as open source, as reported in Sect. 1.3.6.

1.3.6 Software Development and Release

During Year 4, we documented all the developed software. To this end, we prepared dedicated pages on the wiki system of the Consortium. These have been regularly updated after major code changes.

These wiki pages contain instructions on how to *(i)* download to the software, *(ii)* build the code, *(iii)* launch the applications and *(iv)* use the related GUI.

The wiki page in Annex 2.9 presents the instructions on how to test our multi-robot framework via V-REP simulation: it is also available at https://redmine.ciirc.cvut.cz/projects/tradr/wiki/V-REP_Simulation.

Package name	License	Release date	Repo
Path planner + VREP	BSD	After review and related paper submission in arxiv (April 2018)	Will be created in https://github.com/tradr-project
Patrolling	BSD	After review and related paper submission (April 2018)	Will be created in https://github.com/tradr-project
Exploration	BSD	After review and related paper submission (April 2018)	Will be created in https://github.com/tradr-project
PLVS	GPL	After related paper submission (May-June 2018)	To be defined

Table 6: Table of code release.

The wiki page in Annex 2.10 presents the instructions on how to test our multi-robot path planning and patrolling framework: it is also available at https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning.

The above wiki pages are accessible to all the project partners in order to support them during testing and integration.

The instructions on how to compile and use the multi-robot exploration framework are detailed in the README file of the related ROS package.

It is worth noting that all the developed software will be released according to Table 6.

In particular, the path planner, the patrolling and the exploration packages are currently available in the private gitlab system of the consortium. At present time, we are updating the software packages for the preparations of the Year 4 Review. We are going to release the software after the Review, in particular after a submission of the related works (Annex 2.1 and 2.2) to the e-print archive arxiv.org. This is scheduled for April 2018.

As for PLVS, a journal paper is currently under preparation. We are going to release the developed code after a submission of the related work (Annex 2.5) to a journal or to the e-print archive arxiv.org. This is scheduled for May-June 2018.

1.4 Relation to the state-of-the-art

In this section we will describe how the results of WP4 in Year 4 are related to the state-of-the-art.

1.4.1 Multi-robot patrolling

Multi-robot patrolling with advantages of spatial distribution and fault tolerance has found in recent years several applications in real domains where distributed surveillance, inspection or control are crucial (e.g., computer network administration [9, 27], security [3, 5, 49], search and rescue [1, 81, 8], persistent monitoring [98], hotspot policing [19], military [78]).

In this task, a team of robots is required to repeatedly visit a set of areas of interest, in order to monitor them [70, 20, 7, 33, 90, 85].

In the literature two main overlapping taxonomies can be identified for the existing approaches. They provide different classifications of them either on the basis of the kind of application [3, 5] or with respect to the applied theoretical principles [70, 20, 91, 33, 38, 49, 76, 87].

On the basis of the type of application, the existing approaches can be divided in adversarial patrol [110], perimeter patrol [6] and area patrol [85].

Regarding the theoretical baseline, they can be distinguished in pioneer methods [70], graph theory methods [20, 83] and alternative coordination methods [91].

The second taxonomy classifies the state-of-the-art approaches up to 2011 [85]. On the basis of recent research advancements in this field, an alternative subdivision might be devised. A proposal could be to further decompose alternative coordination methods in game theory methods [49], methods resorting to statistical approaches [91, 87], methods using principles from control theory [76] and also logic-based methods [8]. An alternative up-to-date review of some of the aforementioned works can also be found in [87] and in [108].

With respect to the aforementioned taxonomies, the approach that WP4 developed in TRADR is at the intersection of the class of pioneer methods and the class of area patrol.

Scalability and computational complexity constraints and the end-user request of being provided with such a capability motivated the choice the pioneer architecture and the area patrol as type of application, respectively.

Pioneer methods are commonly based on simple architectures where heterogeneous robots with limited perception and communication capabilities are guided to locations that have not been visited for a while, aiming to maintain a high frequency of visits [85]. Under this setting, agents can behave either in a reactive (with local information) or in a cognitive (with access to global information) manner [70, 33]. Over the years, these methods led to what is today better known as *frequency-based patrolling* [20, 34]. In this type of patrolling, the goal of the team of robots is to maximize a given frequency criterion, usually the *idleness* [83, 108], that is, the time between consecutive visits to a particular point within the patrol region [79, 86]. In [82], the authors state that in some cases, simple strategies, like the pioneer ones, with reactive agents, even without communication capabilities,

can achieve equivalent or improved performance when compared to more complex ones. A study of the scalability and performance of some of the patrolling strategies mentioned above has been reported in [85].

Despite the focus that multi-robot patrolling has received recently, it can be noted that there is a manifest lack of practical real-world implementations of such systems [82]. Most of them do not account for 3D[17, 57, 79].

When dealing with real robots operating in harsh environments particular attention has to be paid on the communication, the coordination and the collaboration among the UGVs for safe joint navigation [2, 11, 94].

We improve the current state-of-the-art by proposing an approach, based on a two-level coordination strategy, to cope with the problems rising up in orchestrating a team of real robots deployed in a disaster environment. We migrated multi-robot patrolling on 3D worlds and, finally, we also provide a real deployment with a team of UGVs in an Urban Search & Rescue scenario.

1.4.2 Multi-robot Exploration and Coverage

In the next future, fleets of autonomous robots will be able to flexibly and cooperatively perform multiple tasks such as exploration, coverage and patrolling [58, 84]. Amongst these tasks, an exploration mission is crucial for first assessments and to preliminary build a model of an unknown environment. This operation typically requires a higher level of autonomy and robustness, especially with UGVs or UAVs operating in complex 3D environments [66, 101, 13, 77, 95].

Specifically, the goal of a team of exploring robots is to cooperatively cover an unknown environment with sensory perceptions [39]. Typically, the expected output of an exploration is a 3D map of the environment [101] or the discovery of interesting information/objects [26, 10]. Indeed, multi-robot exploration has a wide range of potential applications, spanning from search-and-rescue operations [22, 73, 63] to surveillance [12], mapping [30] and planetary missions.

In general, a multi-robot system presents many advantages over a single robot system [109]: completion time is reduced, information sharing and redundancy can be used to increase the map accuracy and localization quality [88, 30]. Nonetheless, taking advantage of a multi-robot architecture and actually attaining performance improvements requires the design of sophisticated strategies which can guarantee *cooperation* (avoid useless actions) and *coordination* (avoid conflicts).

In the literature, the problem of covering a scene with sensory perceptions comes in many flavours. Essentially, a sequence of viewpoints (where to gather sensory perceptions) must be planned over a scene of interest. Three main specializations can be considered for this *viewpoint planning* problem: next-best-view, exploration and coverage.

1.4.2.1 Next-best-view

If the scene consists of a single-object without obstacles, *next-best-view* planning algorithms are in order [23, 103, 31, 65]. In this case, the goal is to obtain an accurate 3D reconstruction of an arbitrary object of interest. These approaches typically samples candidate view configurations within a sphere around the target object and select the view configuration with the highest information gain. The downside of these strategies is they do not scale well in multi-objects scenes.

The exploration framework developed in WP4 uses the same philosophy of next-best-view approaches. First, candidate views are generated. In our case, we locally expand a sampling-based tree of candidate views over the traversable terrain surrounding the robot. Then, the next best view is selected. As in [13], we first identify the the branch b^* of the expanded tree which maximizes the total information gain (as in a receding-horizon scheme) and then select as next view the nearby node along b^* .

1.4.2.2 Exploration

When the scene is an unknown environment, viewpoint planning is referred to as *exploration*.

Many exploration strategies are *frontier-based* [107]. Here, the *frontier* is defined as the boundary between known and unknown territory and is approached in order to maximize the expected information gain. In most strategies, a team of robots cooperatively build a metric representation of the environment, where frontier segments are extracted and used as prospective viewpoints.

In [16], Burgard et al. presented a frontier-based decentralized strategy where robots negotiate targets by optimizing a mixed utility function which takes into account the expected information gain, the cost-to-go and the number of negotiating robots. In [52], the same decentralized frontier-based approach is extend to a large-scale heterogenous team of mobile robots.

An incremental deployment algorithm is presented in [51], where robots approach the frontier while retaining visual contact with each other.

In [42, 38], a sensor-based random graph is expanded by the robots by using a randomized local planner that automatically realizes a trade-off between information gain and navigation cost.

A interesting class of exploration methods fall under the hat of *active SLAM* [69, 60, 18, 102] (or integrated exploration [71]), which considers the correlation between viewpoint selection and the onboard localization/mapping quality. For instance, in [64, 41], the utility of a particular frontier region is also considered from the viewpoint of relative robot localization. Similarly, *belief-space planning* methods were proposed to integrate the expected robot belief into motion planning [24, 55].

An interesting multi-robot architecture is presented in [111], where robots are guided through the exploration by a market economy. Similarly, in [97], a centralized frontier-based approach is proposed in which a bidding protocol is used to assign frontier targets to the robots.

In [93], an exploration strategy is driven by the resolution of a partial differential equation. A similar concept is presented in [95]. Here, in order to solve a stochastic differential equation, Shen et al. use particles as an efficient sparse representation of the free space and for identifying frontiers.

Biological-inspired strategies based on Particle Swarm Optimization (PSO) are presented in [25], where an exploration task is defined through the distributed optimization of a suitable sensing function.

In [75], an efficient exploration strategy exploits background information (i.e. a topo-metric map) of the environment in order to improve time performances.

1.4.2.3 Coverage

When a prior 3D model of the scene is assigned, the viewpoints planning problem is known as *coverage* [21, 47]. In [58], a special issue collects new research frontiers and advancements on multi-robot coverage along with multi-robot search and exploration. In [4], Agmon et al. propose a strategy for efficiently building a coverage spanning tree for both online and offline coverage, with aim of minimizing the time to complete coverage. In [48], an efficient frontier-based approach is proposed to solve at the same time both the problems of exploration (maximize the size of the reconstructed model) and coverage (observe the entire surface of the environment, maximize the model completeness). In [43, 36], coverage strategies are presented for inspecting complex structures on the ocean floor by using an autonomous underwater vehicles.

1.4.2.4 Our Approach

In Year 4 we developed a multi-robot exploration framework by casting the two level coordination strategy presented in Annex 2.1 into an 3D exploration context. The resulting distributed technique minimizes and explicitly manages the occurrence of conflicts and interferences in the robot team. In this work, we focused on typical challenges that UGVs must face such as *(i)* avoiding conflicts in narrow passages, *(ii)* performing reliable traversability analysis and coordinated path-planning, *(iii)* reliably localizing in 3D while simultaneously updating and extending the input 3D metric map. In our exploration approach, each robot selects where to scan next by using a receding horizon next-best-view approach [13]. A sampling-based tree is directly expanded on segmented traversable regions of the terrain 3D map. Locations with higher priorities can be online assigned in order to bias the

exploration process. The presented framework can be also used to perform coverage tasks in the case a 3D map of the environment is a priori provided as input.

1.4.3 Efficient Multi-DOF Path Planning

Motion planning for the TRADR UGV has many connections to motion planning for planetary rovers – unstructured terrain, nontrivial robot-terrain interaction (wheel/track slippage etc.) and several degrees of freedom that allow for changing the shape of the robot body (both passive and active) [54, 53, 14]. Howard and Kelly [53] presented an efficient trajectory generation algorithm for wheeled rovers with passive bogie joints where configurations of the robot body in each sampled point are computed using (linearized) forward motion equations on the terrain approximated by a third-order Lagrangian function.

Gianni et al. [44] presented an approach that does not control the robot body, as most path planners do, but instead plans the path of the tips of the two front flipper, computing the required body configuration afterwards using a differential control law on locally linearized terrain.

Several proposed algorithms are based on generating a high-level path in a subset of the configuration space, and having this path, sampling a trajectory along it in the full configuration space [62]. Kim et al. [61] presented an algorithm that automatically “enables” or “disables” some DOFs of the robot during path planning based on the number of failed RRT node expansions.

Other approaches are available for articulated robots, which however only have passive or fixed joints connecting their “legs”. A more or less complex quantity called *traversability index* is computed for each *cell* of the terrain, which is based on simulations of the final robot configuration at the given place [68, 74] This quantity is the provided to the path planner as a part of the cost function.

Ruffi et al. [89] perform search in a highly constrained multidimensional configuration space by using multi-resolution state lattice, which might be seen as a generalization of the algorithms mentioned above.

The importance of having robot orientation as a part of the search space for non-holonomic robots is stressed e.g. by Shin et al. [96].

References

- [1] J. J. Acevedo, B. C. Arrue, J. M. Daz-Bez, I. Ventura, I. Maza, and A. Ollero. Decentralized strategy to ensure information propagation in area monitoring missions with a team of uavs under limited communications. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 565–574, 2013.

- [2] José Joaquín Acevedo, Begoña C. Arrue, Ivan Maza, and Anibal Ollero. A distributed algorithm for area partitioning in grid-shape and vector-shape configurations with multiple aerial robots. *Journal of Intelligent & Robotic Systems*, 84(1):543–557, 2016.
- [3] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345, 2008.
- [4] Noa Agmon, Noam Hazon, Gal A Kaminka, MAVERICK Group, et al. The giving tree: constructing trees for efficient offline and online multi-robot coverage. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):143–168, 2008.
- [5] Noa Agmon, Gal A. Kaminka, and Sarit Kraus. Multi-robot adversarial patrolling: Facing a full-knowledge opponent. *CoRR*, abs/1401.3903, 2014.
- [6] Noa Agmon, Vladimir Sadoy, Gal A. Kaminka, and Sarit Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '08*, pages 55–62. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
- [7] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *ICRA*, pages 1724–1729, 2006.
- [8] Derya Aksaray, Kevin Leahy, and Calin Belta. Distributed multi-agent persistent surveillance under temporal logic constraints this work was partially supported at boston university by the nsf under grants cmmi-1400167 and nri-1426907, and by the onr under grant muri n00014-09-1051. *IFAC-PapersOnLine*, 48(22):174–179, 2015.
- [9] R de C Andrade, Hendrik T Macedo, Geber L Ramalho, and Carlos AG Ferraz. Distributed mobile autonomous agents in network management. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, 2001.
- [10] Alper Aydemir, Andrzej Pronobis, Moritz Göbelbecker, and Patric Jensfelt. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, 2013.
- [11] S. Bereg, L. E. Caraballo, J. M. Díaz-Báñez, and M. A. Lopez. Resilience of a synchronized multi-agent system. *ArXiv e-prints*, April 2016.

- [12] Sourabh Bhattacharya, Salvatore Candido, and Seth Hutchinson. Motion strategies for surveillance. In *Robotics: Science and Systems*, 2007.
- [13] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon” next-best-view” planner for 3d exploration. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1462–1468. IEEE, 2016.
- [14] D. Bonnafous, S. Lacroix, and T. Simeon. Motion generation for a rover on rough terrains. In *IEEE International Conference on Intelligent Robots and Systems. (IROS)*, volume 2, pages 784–789, 2001.
- [15] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):3, 2011.
- [16] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Trans. on Robotics and Automation*, 1(3):376–386, 2005.
- [17] Gonçalo Cabrita, Pedro Sousa, Lino Marques, and A De Almeida. Infrastructure monitoring with multi-robot teams. In *IROS*, pages 18–22, 2010.
- [18] Luca Carlone, Jingjing Du, Miguel Kaouk Ng, Basilio Bona, and Marina Indri. Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent & Robotic Systems*, 75(2):291–311, 2014.
- [19] Huanfa Chen, Tao Cheng, and Sarah Wise. Developing an online cooperative police patrol routing strategy. *Computers, Environment and Urban Systems*, 62:19–29, 2017.
- [20] Y. Chevalereyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 302–308, 2004.
- [21] Howie Choset. Coverage for robotics—a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.
- [22] Francis Colas, Srivatsa Mahesh, François Pomerleau, Ming Liu, and Roland Siegwart. 3d path planning and execution for search and rescue ground robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 722–727. IEEE, 2013.
- [23] Cl Connolly. The determination of next best views. In *Robotics and automation. Proceedings. 1985 IEEE international conference on*, volume 2, pages 432–435. IEEE, 1985.

- [24] Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. Perception-aware path planning. *arXiv preprint arXiv:1605.04151*, 2016.
- [25] Micael S Couceiro, Rui P Rocha, and Nuno MF Ferreira. A novel multi-robot exploration approach based on particle swarm optimization algorithms. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 327–332. IEEE, 2011.
- [26] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. *CoRR*, abs/1711.11543, 2017.
- [27] Timon C. Du, Eldon Y. Li, and An-Pin Chang. Mobile agents in distributed network management. *Commun. ACM*, 46(7):127–132, 2003.
- [28] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. Segmatch: Segment based place recognition in 3d point clouds. pages 5266–5272. IEEE, 2017.
- [29] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. An online multi-robot slam system for 3d lidars. 2017.
- [30] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. An online multi-robot slam system for 3d lidars. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1004–1011. IEEE, 2017.
- [31] Enrique Dunn, Jur Van Den Berg, and Jan-Michael Frahm. Developing visual sensing strategies through next best view planning. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4001–4008. IEEE, 2009.
- [32] M. Freese E. Rohmer, S. P. N. Singh. V-rep: a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.
- [33] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. In *ICRA*, pages 385–390, 2007.
- [34] Yehuda Elmaliach, Noa Agmon, and Gal A. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3):293–320, 2009.
- [35] Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

- [36] Brendan Englot and Franz S Hover. Three-dimensional coverage planning for an underwater inspection robot. *The International Journal of Robotics Research*, 32(9-10):1048–1073, 2013.
- [37] Federico Ferri, Mario Gianni, Matteo Menna, and Fiora Pirri. Point cloud segmentation and 3d path planning for tracked vehicles in cluttered and dynamic environments. In *Proc. of the 3rd IROS Workshop on Robots in Clutter: Perception and Interaction in Clutter*, 2014.
- [38] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Venditelli. The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transaction on Mechatronics*, 14(2):163–175, 2009.
- [39] L. Freda, G. Oriolo, and F. Vecchioli. Exploration strategies for general robotic systems. *DIS Robotics Laboratory Working Paper*, 2009.
- [40] Luigi Freda, Mario Gianni, Fiora Pirri, Abel Gawel, Renaud Dub, Roland Siegwart, and Cesar Cadena. 3d multi-robot patrolling with a two-level coordination strategy - simulations and experiments. *Submitted to Autonomous Robots Journal*.
- [41] Luigi Freda, Francesco Loiudice, and Giuseppe Oriolo. A randomized method for integrated exploration. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2457–2464. IEEE, 2006.
- [42] Luigi Freda and Giuseppe Oriolo. Frontier-based probabilistic strategies for sensor-based exploration. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3881–3887. IEEE, 2005.
- [43] Enric Galceran, Ricard Campos, Narcís Palomeras, David Ribas, Marc Carreras, and Pere Ridao. Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. *Journal of Field Robotics*, 32(7):952–983, 2015.
- [44] Mario Gianni, Federico Ferri, Matteo Menna, and Fiora Pirri. Adaptive Robust Three-dimensional Trajectory Tracking for Actively Articulated Tracked Vehicles*. *Journal of Field Robotics*, 33(7):901–930, 2016.
- [45] H.H. Gonzalez-Banos and J.C. Latombe. Robot navigation for automatic model construction using safe regions. In D. Russ and S. Singh, editors, *Experimental Robotics VII*, volume 271 of *Lecture Notes in Control and Information Sciences*, pages 405–415. Springer, 2000.

- [46] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.
- [47] Noam Hazon and Gal A Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114, 2008.
- [48] Lionel Heng, Alkis Gotovos, Andreas Krause, and Marc Pollefeys. Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1071–1078. IEEE, 2015.
- [49] Erik Hernández, Antonio Barrientos, and Jaime Del Cerro. Selective smooth fictitious play: An approach based on game theory for patrolling infrastructures with a multi-robot system. *Expert Syst. Appl.*, 41(6):2897–2913, 2014.
- [50] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.
- [51] A. Howard, M. Mataric, and S. Sukhatme. An incremental deployment algorithm for mobile robot teams. In *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2849–2854, 2003.
- [52] A. Howard, L.E. Parker, and G. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *Int. J. Robotics Research*, 25(5–6):431–447, 2006.
- [53] Thomas M Howard and Alonzo Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.
- [54] K. Iagnemma, F. Genot, and S. Dubowsky. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 3(May):2286–2291, 1999.
- [55] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *The International Journal of Robotics Research*, 34(7):849–882, 2015.
- [56] Yani Ioannou, Babak Taati, Robin Harrap, and Michael Greenspan. Difference of normals as a multi-scale operator in unorganized point

- p>clouds. In
- 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*
- , pages 501–508. IEEE, 2012.
- [57] L. Iocchi, L. Marchetti, and D. Nardi. Multi-robot patrolling with co-ordinated behaviours in realistic environments. In *IROS*, pages 2796–2801, 2011.
 - [58] Gal A. Kaminka and Amir Shapiro. Editorial: Annals of mathematics and artificial intelligence special issue on multi-robot coverage, search, and exploration. *Annals of Mathematics and Artificial Intelligence*, 52(2):107–108, Apr 2008.
 - [59] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106. IEEE, 2013.
 - [60] Ayoung Kim and Ryan M Eustice. Active visual slam for robotic area coverage: Theory and experiment. *The International Journal of Robotics Research*, 34(4-5):457–475, 2015.
 - [61] Dong Hyung Kim, Youn Sung Choi, Taejoon Park, Ji Yeong Lee, and Chang Soo Han. Efficient path planning for high-DOF articulated robots with adaptive dimensionality. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2015-June, pages 2355–2360, 2015.
 - [62] Tobias Klamt and Sven Behnke. Anytime Hybrid Driving-Stepping Locomotion Planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (September):4444–4451, 2017.
 - [63] Alexander Kleiner, Fredrik Heintz, and Satoshi Tadokoro. Special issue on safety, security, and rescue robotics (ssrr), part 2. *Journal of Field Robotics*, 33(4):409–410, 2016.
 - [64] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3232–3238, 2003.
 - [65] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5031–5037. IEEE, 2011.

- [66] Ivana Kruijff-Korbayová, Luigi Freda, Mario Gianni, Valsamis Ntouskos, Václav Hlaváč, Vladimír Kubelka, Erik Zimmermann, Hartmut Surmann, Kresimir Dulic, Wolfgang Rottner, et al. Deployment of ground and aerial robots in earthquake-struck amatrice in italy (brief report). In *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*, pages 278–279. IEEE, 2016.
- [67] E. Kruse, R. Gutsche, and F.M. Wahl. Efficient, iterative, sensor based 3-D map building using rating functions in configuration space. In *1996 IEEE Int. Conf. on Robotics and Automation*, pages 1067–1072, 1996.
- [68] Philipp Krüsi, Paul Furgale, Michael Bosse, and Roland Siegwart. Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments. *Journal of Field Robotics*, 34(5):940–984, 2017.
- [69] Cindy Leung, Shoudong Huang, and Gamini Dissanayake. Active slam using model predictive control and attractor based exploration. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5026–5031. IEEE, 2006.
- [70] Aydano Machado, Geber Ramalho, Jean-Daniel Zucker, and Alexis Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 155–170. Springer, 2002.
- [71] A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 534–539, 2002.
- [72] M. Menna, M. Gianni, F. Ferri, and F. Pirri. Real-time autonomous 3d navigation for tracked vehicles in rescue environments. In *IROS*, pages 696–702, 2014.
- [73] Robin R Murphy. *Disaster robotics*. MIT press, 2014.
- [74] Mohammad Norouzi, Jaime Valls Miro, and Gamini Dissanayake. Planning stable and efficient paths for articulated mobile robots on challenging terrains. In *Australasian Conference on Robotics and Automation, ACRA*, 2013.
- [75] Stefan Oßwald, Maren Bennewitz, Wolfram Burgard, and Cyrill Stachniss. Speeding-up robot exploration by exploiting background information. *IEEE Robotics and Automation Letters*, 1(2):716–723, 2016.

- [76] D. Panagou, D. M. Stipanovi, and P. G. Voulgaris. Distributed coordination control for multi-robot networks using lyapunov-like barrier functions. *IEEE Transactions on Automatic Control*, 61(3):617–632, 2016.
- [77] Christos Papachristos, Shehryar Khattak, and Kostas Alexis. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4568–4575. IEEE, 2017.
- [78] Chul-Hwan Park, Yeong-Dae Kim, and BongJoo Jeong. Heuristics for determining a patrol path of an unmanned combat vehicle. *Comput. Ind. Eng.*, 63(1):150–160, 2012.
- [79] F. Pasqualetti, J. W. Durham, and F. Bullo. Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. *IEEE Transactions on Robotics*, 28(5):1181–1188, 2012.
- [80] Trung T. Pham, Markus Eich, Ian Reid, and Gordon Wyeth. Geometrically Consistent Plane Extraction for Dense Indoor 3D Maps Segmentation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, October 2016.
- [81] C. Pippin and H. Christensen. Trust modeling in multi-robot patrolling. In *ICRA*, pages 59–66, 2014.
- [82] D. Portugal and R. P. Rocha. Scalable, fault-tolerant and distributed multi-robot patrol in real world environments. In *IROS*, pages 4759–4764, 2013.
- [83] David Portugal and Rui Rocha. Msp algorithm: Multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1271–1276, New York, NY, USA, 2010. ACM.
- [84] David Portugal and Rui Rocha. A survey on multi-robot patrolling algorithms. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pages 139–146. Springer, 2011.
- [85] David Portugal and Rui P. Rocha. Multi-robot patrolling algorithms: examining performance and scalability. *Advanced Robotics*, 27(5):325–336, 2013.
- [86] David Portugal and Rui P. Rocha. *Retrieving Topological Information for Mobile Robots Provided with Grid Maps*, pages 204–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

- [87] David Portugal and Rui P. Rocha. Cooperative multi-robot patrol with bayesian learning. *Autonomous Robots*, 40(5):929–953, 2016.
- [88] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, 2001.
- [89] Martin Ruffi, Dave Ferguson, and Roland Siegwart. Smooth path planning in constrained environments. *2009 IEEE International Conference on Robotics and Automation*, pages 3780–3785, 2009.
- [90] Tiago Sak, Jacques Wainer, and Siome Klein Goldenstein. *Probabilistic Multiagent Patrolling*, pages 124–133. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [91] Hugo Santana, Geber Ramalho, Vincent Corruble, and Bohdana Ratitch. Multi-agent patrolling with reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '04, pages 1122–1129. IEEE Computer Society, 2004.
- [92] Max Schwarz. nimbro_network - ROS transport for high-latency, low-quality networks. https://github.com/AIS-Bonn/nimbro_network, 2017. [Online; accessed 20-February-2017].
- [93] Robbie Shade and Paul Newman. Choosing where to go: Complete 3d exploration with stereo. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2806–2811. IEEE, 2011.
- [94] M. Shahriari and M. Biglarbegian. A new conflict resolution method for multiple mobile robots in cluttered environments with motion-liveness. *IEEE Transactions on Cybernetics*, PP(99):1–12, 2016.
- [95] Shaojie Shen, Nathan Michael, and Vijay Kumar. Stochastic differential equation-based exploration algorithm for autonomous indoor 3d exploration with a micro-aerial vehicle. *The International Journal of Robotics Research*, 31(12):1431–1444, 2012.
- [96] Seho Shin, Joonwoo Ahn, and Jaeheung Park. Desired orientation RRT (DO-RRT) for autonomous vehicle in narrow cluttered spaces. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2016-November, pages 4736–4741, 2016.
- [97] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *17th AAAI/12th IAAI*, pages 852–858, 2000.

- [98] Cheng Song, Lu Liu, Gang Feng, and Shengyuan Xu. Optimal control for multi-agent persistent monitoring. *Automatica*, 50(6):1663–1668, 2014.
- [99] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1):137–147, 2014.
- [100] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [101] Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3-4):181–198, 2003.
- [102] Rafael Valencia and Juan Andrade-Cetto. Active pose slam. In *Mapping, Planning and Exploration with Pose SLAM*, pages 89–108. Springer, 2018.
- [103] J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. Volumetric next-best-view planning for 3d object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 11(10):159, 2014.
- [104] Aisha Walcott-Bryant, Michael Kaess, Hordur Johannsson, and John J Leonard. Dynamic pose graph slam: Long-term mapping in low dynamic environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1871–1878. IEEE, 2012.
- [105] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous: Spatially extended kinectfusion. *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.
- [106] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.
- [107] B. Yamauchi. Frontier-based exploration using multiple robots. In *2nd Int. Conf. on Autonomous Agents*, pages 47–53, 1998.

- [108] Chuanbo Yan and Tao Zhang. Multi-robot patrol: A distributed algorithm based on expected idleness. *International Journal of Advanced Robotic Systems*, 13(6):1729881416663666, 2016.
- [109] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.
- [110] R. Yehoshua, N. Agmon, and G. A. Kaminka. Robotic adversarial coverage: Introduction and preliminary results. In *IROS*, pages 6000–6005, 2013.
- [111] R. Zlot, A. Stenz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *2002 IEEE Int. Conf. on Robotics and Automation*, pages 3016–3023, 2002.

2 Annexes

2.1 Freda (2017), “3D Multi-Robot Patrolling with a Two-Level Coordination Strategy: Simulations and Experiments”

2.1.0.1 Bibliography

Freda, L. Gianni, M. Pirri, F. Dubé ,R. Gawel, A. Cadena, C. and Siegwart, R. “3D Multi-Robot Patrolling with a Two-Level Coordination Strategy: Simulations and Experiments”. Submitted to Autonomous Robots. Springer. New revised version after first review.

2.1.0.2 Abstract

Teams of robots patrolling harsh and complex environments can experience interference and spatial conflicts crucially affecting their activity. If neglected, these fundamental aspects can hinder patrolling methods to attain sound performances. In this work we present a distributed multi-robot patrolling technique in which a two-level coordination strategy is used in order to minimize and explicitly manage the occurrence of conflicts and interferences. On a first level, a topologically based strategy, under which each agent selects its target node on a topological map, relies on a shared heuristic criterion and a coordination mechanism so as to prevent topological conflicts, such as same target node selection. The second level hosts strategies based on a metric representation of space, using a laser-based SLAM system, in which each robot path planner manages and minimizes the occurrences of spatial conflicts applying a multi-robot traversability function. The approach is fully distributed and inherently fault-tolerant. Extensive simulations and experiments show how the proposed method can effectively and efficiently take care of conflicts and interferences amid robots in a patrolling team.

2.1.0.3 Relation to WP

This work presents research advancements with respect to cooperation and coordination, at different architectural levels, for a team of UGVs patrolling a real disaster scenario.

2.1.0.4 Availability

Restricted. Not included in the public version of this deliverable.

2.2 Freda (2018), “3D Multi-Robot Exploration with a Two Level Coordination Strategy and Prioritization”

2.2.0.5 Bibliography

Freda, L. Pirri, F. “3D Multi-Robot Exploration with a Two Level Coordination Strategy and Prioritization”. Work in progress.

2.2.0.6 Abstract

This work present a 3D multi-robot exploration framework for a team of UGVs moving on unknown harsh terrains. The framework was designed by casting the two level coordination strategy presented in [40] into the context of multi-robot exploration. The resulting distributed exploration technique minimizes and explicitly manages the occurrence of conflicts and interferences in the robot team. Each robot selects where to scan next by using a receding horizon next-best-view approach [13]. A sampling-based tree is directly expanded on a traversable 3D map of the terrain. Locations with higher priorities can be online assigned in order to bias the exploration process. The presented framework can be also used to perform coverage tasks in the case a map of the environment is a priori provided as input.

2.2.0.7 Relation to WP

This work presents research advancements with respect to cooperation and coordination, at different architectural levels, for a team of UGVs exploring a real disaster scenario.

2.2.0.8 Availability

Restricted. Not included in the public version of this deliverable.

2.3 Freda (2017), “3D Multi-Robot Exploration and Coverage with a Receding Horizon Next-Best-View Approach”

2.3.0.9 Bibliography

Freda, L. and Pirri, F. “3D Multi-Robot Exploration and Coverage with a Receding Horizon “Next-Best-View”. TRADR Techday, November 2017.

2.3.0.10 Abstract

This poster was presented at the TRADR Techday, which was held in Rotterdam on November 17 2017. It presents the 3D multi-robot exploration framework implemented in the TRADR system.

2.3.0.11 Relation to WP

This work presents research advances and the implementation of a 3D multi-robot exploration framework within WP4.

2.3.0.12 Availability

Unrestricted.

2.4 Freda (2017), “3D Multi-Robot Patrolling with a Two-Level Coordination Strategy”

2.4.0.13 Bibliography

Freda, L. Gianni, M. Pirri, F. Dubé, R. Gawel, A. Cadena, C. and Siegwart, R. “3D Multi-Robot Patrolling with a Two-Level Coordination Strategy”. TRADR Techday, 2017-2018.

2.4.0.14 Abstract

This poster was presented at the TRADR Techday, which was held in Rotterdam on November 17 2017. It presents the 3D multi-robot patrolling framework implemented in the TRADR system.

2.4.0.15 Relation to WP

This work presents the research advancements with respect to the coordination and collaboration at different architectural level of a team of UGVs patrolling a real disaster scenario, under the supervision of the end-users.

2.4.0.16 Availability

Unrestricted.

2.5 Freda (2018), “PLVS: An Open-Source RGB-D SLAM System with Keypoints, Keylines, Volumetric Mapping and 3D Incremental Segmentation”

2.5.0.17 Bibliography

Freda, L. Spinelli, I. Pirri F. “PLVS: An Open-Source RGB-D SLAM System with Keypoints, Keylines, Volumetric Mapping and 3D Incremental Segmentation”. In preparation. To be soon submitted to Journal of Robotics Research (or similar).

2.5.0.18 N.B.

We have two versions of this paper:

- (1) a *short version*, which was submitted to NVIDIA for an NVIDIA GPU Grant Request. The request was approved on December 30th and NVIDIA was happy to support our work with the donation of a Jetson TX2 board.
- (2) A *long version*, which is currently in preparation for a journal submission.

2.5.0.19 Abstract

We presents PLVS: a real-time system which leverages sparse RGB-D SLAM, volumetric mapping and 3D unsupervised incremental segmentation. The system runs entirely on CPU. The SLAM module is keyframe-based and relies on the extraction and tracking of sparse points and segment lines as features. Volumetric mapping runs in parallel with respect to the SLAM front-end and allows to obtain a 3D reconstruction of the explored environment by fusing point clouds backprojected from keyframes. Different volumetric mapping methods are supported and integrated in PLVS. A novel reprojection error is used for bundle-adjusting line segments. This error takes advantage of depth information in order to better stabilize the position estimate of line segment endpoints and can be also used with stereo camera systems. An incremental segmentation method is implemented and integrated in the PLVS framework. We present both qualitative and quantitative evaluations of the PLVS framework on publicly available datasets. The software is available as open-source.

2.5.0.20 Relation to WP

This work presents research analysis for dense reconstruction and segmentation of points clouds. This could enable a reliable traversability analysis, accurate and coloured maps for supporting a robust autonomous UGV navigation. This work is at the intersection of WP1, WP2 and WP4.

2.5.0.21 Availability

Restricted. Not included in the public version of this deliverable.

2.6 Brandizzi (2017), “Comparison between Segmentation Methods for Point Clouds”**2.6.0.22 Bibliography**

Brandizzi, N. “Comparison between Segmentation Methods for Point Clouds”. **Bachelor Thesis**. University of Rome “La Sapienza”, ALCOR Lab.

2.6.0.23 Abstract

Object clustering and segmentation has been a long time studied argument in computer vision and image processing, its application range from video surveillance to domestic object recognition. With the constant progress of computers and their computational speed, some research areas in robotics and computer vision started to progressively aim to new and faster artificial intelligence approaches to identify obstacles and estimate new routes to move from a starting point A to a goal point B (piano movers problem). In this thesis, we want to address and compare some of the most important algorithms for plane segmentation and point cloud clustering which constitute a fundamental basis in the analysis of 3D point clouds for task such a robotic navigation. In particular we focus on the problem of stairs segmentation which consists in the recognition of a set of steps lying on a the main oblique plane. Each examined algorithm is presented along with its state-of-art implementation and combined with other multiple edge extraction methods.

2.6.0.24 Relation to WP

This research and engineering work presents the analysis we conducted on geometric-based point cloud segmentation and clustering methods. These are crucial ingredients in terrain traversability analysis for UGV navigation. This work is at the intersection of WP1, WP2 and WP4.

2.6.0.25 Availability

Restricted. Not included in the public version of this deliverable.

2.7 Wiki-MR-Use-Cases (2017), “Review Yr3 Recommendations for WP4”

2.7.0.26 Bibliography

Freda, L. “Review Yr3 Recommendations for WP4.”. 2017-2018.

2.7.0.27 Abstract

This wiki page presents: *(i)* the recommendations we received from Reviewers on WP4 (Multi-Robot Collaboration) *(ii)* a concise and intuitive description of the implemented multi-robot algorithms, as a basis for discussion and *(iii)* the interactions with End-Users and their required feedback

2.7.0.28 Relation to WP

It is related to the addressing of Reviewers' comments of WP4 (see Section 1.2).

2.7.0.29 Availability

Unrestricted. Included in the public version of this deliverable. Also available at https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Review_Yr3_Recommendations.

2.8 Wiki-MR-Use-Cases (2017), “Multi-Robot Use Cases Wiki page on Redmine”

2.8.0.30 Bibliography

Freda, L. “Multi-Robot Use Cases Wiki page on Redmine.”. 2016-2017-2018.

2.8.0.31 Abstract

Wiki page on Redmine collecting useful information for the identification of interesting multi-robot tasks within the TRADR project. The objective of this page is two-fold: (1) to clearly define a set of doable tasks which will be actually implemented for the reviews and demos and (2) to foster collaboration and discussions within the TRADR team.

2.8.0.32 Relation to WP

It is related to the contribution of WP4 to use-cases of Year 3 of TRADR project (see Section).

2.8.0.33 Availability

Unrestricted. Included in the public version of this deliverable. Also available at https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Multi-Robot_Use_Cases_Definition.

2.9 Wiki-VREP (2017), “V-REP Simulation Wiki page on Redmine”

2.9.0.34 Bibliography

Freda, L. “V-REP Simulation Wiki page on Redmine.”. 2016-2017.

2.9.0.35 Abstract

Wiki page on Redmine describing how to make up and running in V-REP the TRADR system.

2.9.0.36 Relation to WP

It is related to the implementation of the research work presented in Sub-section 1.3.6, Section 1.3.

2.9.0.37 Availability

Unrestricted. Included in the public version of this deliverable. Also available at https://redmine.ciirc.cvut.cz/projects/tradr/wiki/V-REP_Simulation.

2.10 Wiki-MR (2017), “Multi-Robot Path Planning and Patrolling Wiki page on Redmine”

2.10.0.38 Bibliography

Freda, L. “Multi-Robot Path Planning and Patrolling Wiki page on Redmine.”. 2016-2017.

2.10.0.39 Abstract

Wiki page on Redmine describing how to get access to the code related to the implementation of the multi-robot path planning and patrolling. It contains a detailed set of guidelines describing how to set up the application in a virtual simulated environment as well as how to use it.

2.10.0.40 Relation to WP

It is related to the implementation of the research work presented in Sub-section 1.3.6, Section 1.3.

2.10.0.41 Availability

Unrestricted. Included in the public version of this deliverable. Also available at https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning.



Long-Term Human-Robot Teaming for Robot Assisted Disaster Response



3D Multi-Robot Exploration and Coverage with a Receding Horizon “Next-Best-View” Approach

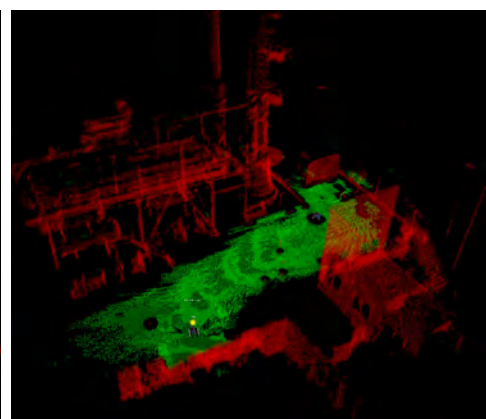
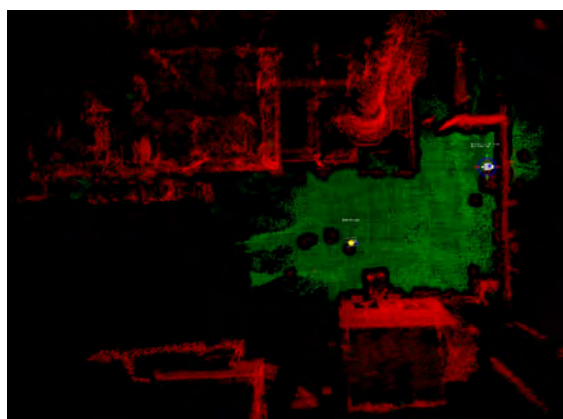
Luigi Freda and Fiora Pirri

Alcor Laboratory, DIAG “A. Ruberti”, Sapienza University of Rome

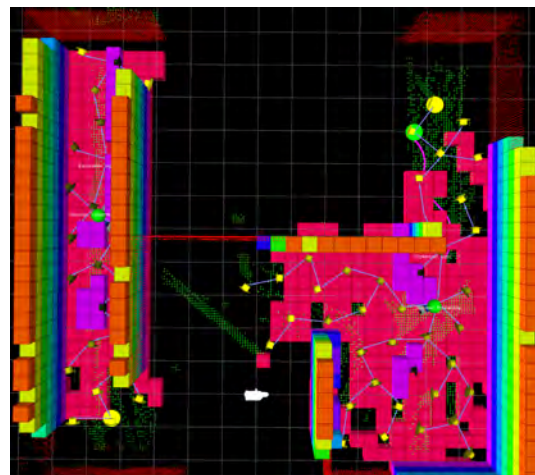
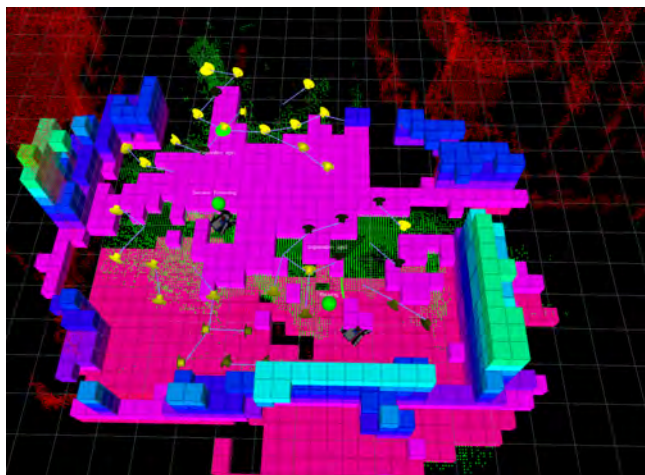
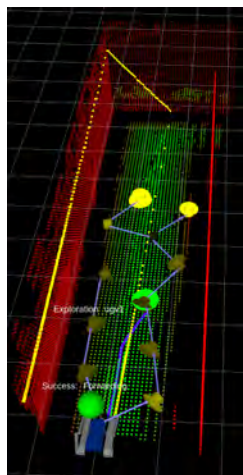


Exploration mission: cover with sensory perceptions (3D laser scans) an **unknown environment** in a fast and robust way.

Coverage mission: a map of the environment is already available and provided to the robots; the UGVs autonomously plan their paths in order to cover the environment with sensory perceptions.

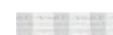
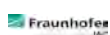


Challenge: design an efficient coverage/exploration algorithm which (i) can run on the robots onboard computers in a distribute and online fashion (ii) can robustly guide the robot team in an harsh environment with limited communication bandwidth and (iii) can minimize interference and spatial conflicts.



Contribution: an efficient framework which allows both coverage and exploration in distribute fashion. It is designed so as to maximize within a prefixed range the information gain and so as to guarantee both coordination (avoid conflicts/collisions) and collaboration (avoid inefficient actions). Terrain traversability is analyzed and used to steer the robots exploration towards safe paths. For coverage tasks, a map of the environment can be loaded by the robot at the beginning of the mission.

Approach: the proposed method is based on a receding horizon “next-best-view” approach. A tree is expanded on the traversable terrain by using a randomized A*. Different utility function can be used to bias the exploration toward interesting regions. The best branch of the tree is computed so as to maximize the amount of unmapped space that can be explored/covered. Multi-robot traversability is used with the aim of minimizing spatial conflicts.





Long-Term Human-Robot Teaming for Robot Assisted Disaster Response

3D Multi-Robot Patrolling with a Two-Level Coordination Strategy

Luigi Freda¹, Mario Gianni¹, Fiora Pirri¹,

Renaud Dube², Abel Gawel², Cesar Cadena², Roland Siegwart²

¹Alcor Laboratory, DIAG "A. Ruberti", Sapienza University of Rome

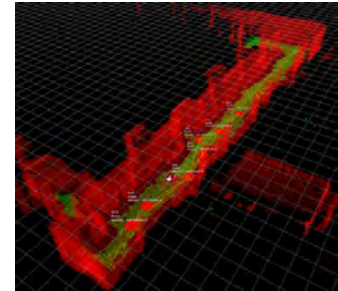
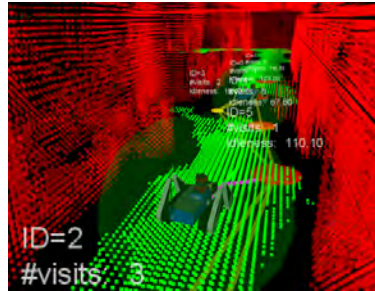
²Autonomous System Laboratory, ETH Zurich



SAPIENZA
UNIVERSITÀ DI ROMA

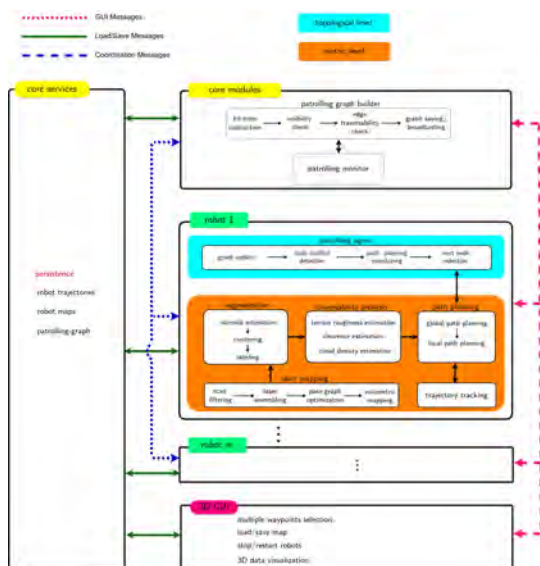
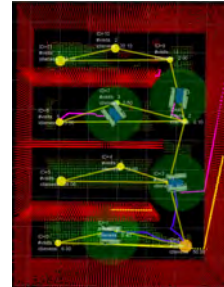
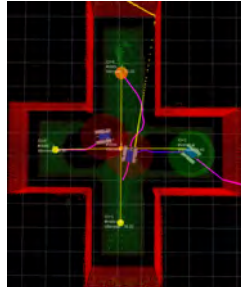
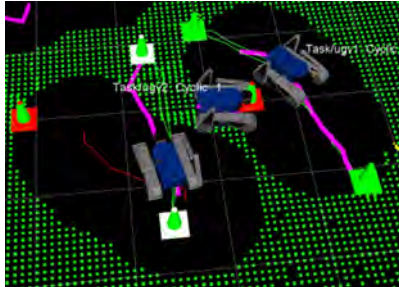
ALCOR Lab

Patrolling mission: robots are required to continuously visit some points of interest so as to maximize the visit frequency of each point. In an automated surveillance system, the robot stops at each goal station and analyses the scene searching for victims, scene variations or abandoned/removed objects.

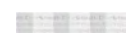
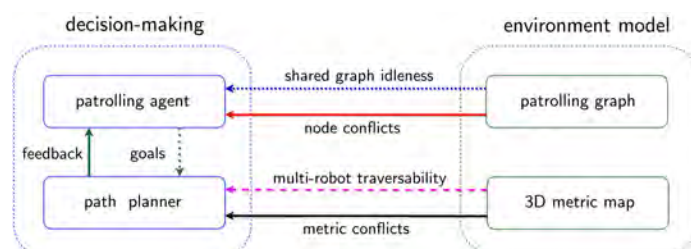


Challenge: teams of robots patrolling harsh and complex environments can experience **interference** and **spatial conflicts** one another, which crucially affect their activity. Neglecting the occurrence of these events hinders both soundness and reliability of a patrolling process.

Contributions: a distributed multi-robot patrolling technique, which uses a two-level coordination strategy that minimizes and explicitly manages the occurrence of conflict and interference.



Two-level approach: the first level guides the agents to single out exclusive target nodes on a **topological map**. This selection relies on a shared heuristic criterion and a coordination mechanism preventing topological conflicts. The second level hosts strategies based on a **metric representation** of space and it is supported by a laser-based SLAM system. Here, each robot path planner negotiates spatial conflicts by applying a multi-robot traversability function. Continuous interactions between these two levels ensure coordination and conflict resolution. The presented method is fully distributed and inherently fault-tolerant.



Review Yr3 Recommendations for WP4

This page presents:

- the recommendations we received from Reviewers on WP4 (Multi-Robot Collaboration)
- a concise and intuitive description of the implemented multi-robot algorithms, as a basis for discussion
- the interactions with End-Users and their required feedback

Reviewers' Recommendations

Main recommendations:

- "we recommend a more explicit link between requirements defined by end users on patrolling and the specific patrolling solution chosen for the TRADR system"
- "clarify this design option in Yr4 deliverables of WP4 to better link requirements with the implemented solution."

Required actions:

- discuss with end-users about the optimization criteria that drive the current implementations of patrolling and exploration
- collect feedback about these optimization criteria from end-users

In order to satisfy Reviewers' requests, we [ROME] pose the following **main questions** to end-users:

- please, read the patrolling criteria descriptions below and "our patrolling visit frequency criteria": is this reasonable for you?
- please, read the exploration and coverage descriptions below: would you use the proposed prioritized exploration?

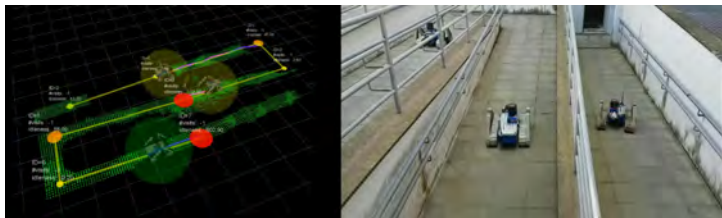
In the following sections, you can find some brief descriptions of the input data and the main objectives that characterize our current implementations of patrolling, exploration and coverage.

The used **terminology** is taken from the Robotics literature and, for simplicity, we kindly invite end-users to adopt it.

Patrolling

This is a very concise and intuitive description of some basic patrolling principles.

- **input**: the robot team is assigned a 3D map of a **known environment** and a **patrolling graph** (nodes=points of interest; edges=traversable paths)
 - on each node there is a "balloon" whose size represents the time the node has been left unvisited by robots (its "idleness"); the balloon keeps on inflating as time passes by, i.e., the bigger the balloon the longer the node has been left unvisited by robots
 - when a robot reaches a node the balloon of that node is completely deflated
- **objective**: the robots collaborate in order to continuously visit all the nodes and keep their balloons as small as possible; this results in **maximizing the visiting frequency of each assigned node**; in other words, we want to maintain each node unvisited for the shortest time possible by revisiting it again and again (in the meanwhile, when a node is reached, we check if something new happened there or in its surroundings)
- our patrolling page: <https://sites.google.com/a/dis.uniroma1.it/3d-cc-patrolling/>



Different types of patrolling criteria:

- **frequency-based**: guarantee a high frequency of visits in each part of the designated patrol area
- **adversarial**: move efficiently in the patrol area in order to discover possible intruders (adversaries which try to penetrate through the patrol area without being detected)

For frequency-based patrolling, the most common **visit frequency criteria** (according to which patrol algorithms can be **evaluated**) are the following ones:

(see the papers "A Realistic Model of Frequency-Based Multi-Robot Polyline Patrolling" and "Multi-Robot Area Patrol under Frequency Constraints" by Y. Elmaliach, A. Shiloni, and G.A. Kaminka)

1. **Uniform frequency**: The goal is to decrease the variance between the visit frequencies of the nodes i.e., all targets should ideally be visited with uniform frequency f .
2. **Average frequency**: In the case where uniformity cannot be guaranteed, the goal is to increase the average frequency f in which targets are visited.

3. **Under-bounded frequency:** The goal is to increase the minimum frequency at which any target is visited, such that every target is visited with a frequency of at least $1/f$. In other words, all targets should be monitored at least once every f cycles.

Implemented patrolling visit frequency criteria: our current patrolling implementation is frequency-based and puts the accent on the average visiting frequency. Each robot selects as new target node the closest one with the highest idleness. In general, it is very difficult to guarantee a uniform frequency or enforce an under-bound frequency in a **dynamic** environment (where obstacles can move and teammates can interfere and obstruct one another) with narrow passages and a **complex topology** (i.e. in our case, we do not have a simple closed fence but many POIs which are distributed over the environment at need by the end-user). In general, uniform frequency and under-bound frequency criteria are considered when patrolling a fence/perimeter (not an area) which is simpler than patrolling an area with complex topology.

[IKK]: The reviewers' recommendation concerns the question what the patrolling algorithm optimizes for, i.e., what it maximizes (or minimizes). In the present version it maximizes visiting frequency (minimizes "idleness"). The reviewers were asking whether this corresponds to end-user requirements. Maybe there are other patrolling/monitoring schemes that would make (better) sense to them? For example: visit node A at regular intervals once an hour, visit node B at regular intervals every half an hour. ...

[Luigi]: the current patrolling implementation aims at maximizing the visiting frequency of all the nodes (the accent is on the average frequency). As explained above: it is very difficult to guarantee a uniform frequency in a complex dynamic environment (not a simple fence!) where obstacles can move and teammates can interfere and obstruct one another.

Exploration

This is a very concise and intuitive description of some basic exploration principles.

- *input:* none; the robots wake up in a completely **unknown environment**;
- *objective:* cover the unknown environment with sensory perceptions in order to **build a 3D map**; we expect the task to be completed in the most "efficient" way (usually minimizing completion time)
- *a simple analogy:* imagine the environment is enclosed in a big box; this box is completely filled with a liquid which after a while becomes solid; every time a robot switches on the laser (to acquire a new 3D scan), the laser carves a 3D cone inside the solid matter; the final objective of the exploration is to clean all the environment from the solid matter by using laser 3D scans;
- *process:* the exploration is performed through a simple loop:
 1. acquire a new 3D scan (clean a small portion of environment)
 2. plan the next scan position within the currently "cleaned"(explored) region in order to "clean" as much new space as possible from the new planned scan position
 3. move towards the newly selected scan position
 this loop implements what is considered a "next best view" strategy (an online technique)
- **prioritized exploration:** during the exploration process, robots can be assigned some Points Of Interest (*POI*s) with a higher priority; in this case, the robots immediately get as close as possible to these POIs and the explore their surroundings to gather more information; for example, if victims are detected and localized by the robots, end-users can assign a higher priority to their locations.

Implemented exploration: for sake of clarity, the implemented exploration process is more involved than described above; the method is based on a receding horizon "next-best-view" approach. At each step, a tree is expanded on the currently traversable terrain. Different utility functions can be used to bias the exploration toward interesting regions. The best branch of the tree is computed so as to maximize the amount of unmapped space that can be explored/covered. Multi-robot traversability is used with the aim of minimizing spatial conflicts.

Coverage

This is a very concise and intuitive description of some basic exploration principles.

- *input:* the robots are given a full map of the environment (e.g. a point cloud modeling at best the full environment)
 - *objective:* cover the **known environment** with sensory perceptions; possible tasks: update the 3D map, discover changes in the environment, clean the environment (vacuum cleaners), etc...
 - *main difference with respect to exploration:* here, each robot is not constrained to select its next target position within the explored region; the next target position can be selected all over the map of the known environment
 - prioritizing: POIs with higher priorities can be assigned by end-users
 - indeed, robot vacuum cleaners perform coverage :-)
- [IKK]: really? the roomba we used to use (the first model) had no map (and was not building one either); maybe that's changed since then ;-)*
- [Luigi]: at present time, most advanced vacuum cleaners are able to first map the environment by using SLAM techniques and then perform an appropriate strategy (off-line or online) for covering the environment*

[IKK] Would end-users employ a prioritization strategy or would they prefer a more systematical "coverage" of the area? (This is an important aspect to ask the end-users about.)

[IKK]: Are there "fixed" strategies for covering an area, e.g., going from left to right in a particular pattern?

[Luigi]: off-line strategies (i.e. strategies which are pre-computed before starting the mission) can be only used for coverage (not for exploration!). You need to a priori know a model of the environment in order to pre-plan some actions. For coverage

tasks, a famous off-line approach is the boustrophedon decomposition which is an exact decomposition of the environment in cells, where each cell in the boustrophedon is covered with simple back and forth motions. Again, such techniques can be only applied for coverage tasks since in order to decompose the environment you need to know it a priori. The main downside of off-line approaches is that they usually assume the environment is static (obstacles do not move) and require exact action executions. The exploration is commonly performed by using the aforementioned "next best view strategy": this is an **online** approach which can best face the compelling uncertainty in perceptions, modeling and action executions.

Feedback From End-Users

[Vigili del Fuoco] Emanuele Gissi:



- Emanuele and Luigi discussed about this page content in a dedicated phone call.
- [Emanuele's conclusions after the phone-call]: Dear Luigi, as requested, we analyzed the different frequency-based **patrolling** criteria. In general, these criteria can be used to evaluate the performance of a patrolling system. Even if "uniform frequency" and "under-bounded frequency" seem reasonable criteria, in real operative scenarios the environments usually present a complex topology and dynamic changes may occur. In these contexts, optimizing the average frequency appears the best realistic compromise for a patrolling system. The suggested prioritization scheme for **exploration** seems very promising. The possibility to steer the exploration towards interesting locations provides a useful capability in the direction of an actual inspection. This feature can be actually used in operative scenarios, where victims or dangerous objects can be detected and may require a robot to readily perform an on-spot visit. This is why I consider your work as a good fit of our current needs.

[GB] Dominic van de Velde:

- [Luigi] Question1: is our current **patrolling** implementation reasonable for you? Please consider that we selected the optimization of the average visit frequency as our patrolling criteria since, in our view, this is the best and most practical compromise in the real world. [Domenic] I think the patrolling with the nodes is a good tool. The balloons that grow in time gives a good view for the operator. What I don't see/read is the possibility of prioritize the POI's (nodes). Is it for the operator possible to give different priorities or time-tables to the different PIO's? I watched the patrolling page, it is good to see it work online!
- [Luigi] Question2: do you think is useful to have such a prioritized **exploration**? [Domenic] I think it's a need-to-have option. The operator / end-user must be able to change the priority of the exploration. It's good that the robot starts with the exploration and make his own choices (based on knowledge, information and protocols). But if there is a specialness in the area the operator/end-user must be in control to change the priority.

[FDDO] Norbert Pahlke:

- Patrolling:** I agree with the colleagues concerning the question of the reviewers. Depending on the situation it should be possible to prioritize nodes/areas for patrolling routines and assign them a different patrolling strategy or the system itself determines the best strategy by information about e.g. the different time needed for the different distances, number of available UGVs, ... how ever. Typically areas with the highest chance of changes are more interesting than the other ones. A promising approach is the average frequency, because it offers the flexibility regarding the requirement given above.
- Exploration:** a prioritized exploration is necessary because of several reasons but it depends also on its execution, which is in turn depending on environment conditions and additional sensors like a RGB/IR camera. The operator should give priority to special places like assumed victims or danger, but he is also a guide to lead the robot step by step through the unknown area on the basis of images and his natural cognitive situational awareness. The operator may receive information from other sources and would change and determine a new aim. That must be possible. It would be a requirement that a UGV can choose the best and shortest way to a place across the known map.

 (571 KB)  Freda, Luigi, 01/10/2018 10:54 AM

[Wiki](#) » [Discussions](#) »

Multi-Robot Use Cases

The **goals** of this page are:

1. to collect useful information for the identification of interesting multi-robot tasks within the TRADR project
2. to clearly define a set of **doable tasks** which will be actually **implemented** for the reviews and demos
3. to collect all the related implementation details and issues that will arise along the way
4. foster collaboration and discussions within the TRADR team on multi-robot collaboration topics

End-users are kindly invited to add relevant information and define the desired tasks.

The **companion wiki page** [Review_Yr3_Recommendations](#) for WP4 reports:

- the recommendations we received from Reviewers on WP4
- a concise and intuitive description of the implemented multi-robot algorithms, as a basis for discussion
- the interactions with End-Users and their required feedback

Basic Assumptions

Robots deployment

- *Ground layer*: two UGVs collaborate on the ground level in order to perform the assigned cooperative tasks (exploration, patrolling, etc..)
- *Air layer*: one UAV independently flies over the facility in order to continuously monitor it and provide a strategic overview

End-users General Requirements

End-users are kindly invited to add relevant requirements for the identification/selection of the multi-robot tasks which will be actually implemented.

FDDO (Norbert)

- common monitoring
- common search in a building keeping connection between each other and to the command car or an other human -> relay function, swarm behaviour
- common exploration of a special dangerous area as fast as possible, UGVs alone and/or in collaboration with the UAVs
- common exploration by UAVs

Questions

Luigi: what do you mean by *monitoring*? Does it match with the patrolling task defined below?

[Vlada]: I guess those are just synonyms; my impression is that Norbert would like to see robots moving around and monitoring things like fire, gas, or major changes in the mission area (blocked corridor for example)

[Norbert]: patrolling means you look around for changes, dangers ... but more or less without concrete objectives. Monitoring means the other way around you have fixed places where you take measurements, ...

[Luigi]: in my view, Norbert's patrolling definition can be implemented as an exploration or coverage task as defined below (see sect. *Low-Level Mission/Task Definition*), depending on the availability of a previously built map, and should be combined with a parallel detection module (which is responsible for instance of detecting victims or other specific objects of interest). Norbert's monitoring definition can be implemented as a patrolling as defined below (see sect. *Low-Level Mission/Task Definition*). We took the definitions below from robotics literature.

High-Level Mission/Task Definition

Here some **conceptual** definitions of the multi-robot collaboration tasks.

- **Use case specifications:**
 - a fire/accident causes the discharge of dangerous substances into the air -> UGVs get the order for monitoring, persistent data of environment + data about weather = autonomous planning and measuring -> value driven measurement concerning path planning
 - a building of "second priority" could be explored by UGVs which relieves the In-field rescuer/Attack Teams (after reorganisation)
 - the exploration of an unknown, may be damaged area, should be done quick. UAVs deliver Lidar data for a predictive UGV path planning -> Lidar mapping (discussed in Zurich) -> systematologies of area exploration
 - time critical beginning of a mission: UAVs split the job into parallel executable tasks 1) overview, 2) victim search, 3) 3D model and further images, 4) other risks (fire, substances, ...), 5) monitoring -> UGVs for details
- **Human interface definition:**
 - end-user selects interesting areas for patrolling in RVIZ (is that possible?) *Luigi: this is already implemented in RVIZ*
 - message passing through memory (StarDog)

Low-Level Mission/Task Definition

Here we define a small collection of low-level multi-robot tasks which can be relevant within the TRADR project:

- **Off-line coverage:** *cover* (with sensory perceptions) an environment which is a priori known in the most "efficient" way.
- **Patrolling:** an environment must be continually surveyed by a group of robots such that each point is visited/covered with equal frequency - given a *graph-representation* of the environment (*nodes* represent reachable and safe regions, *edges* represent traversable paths joining them), patrolling requires continuously visiting all the graph nodes so as to minimize the time lag between two visits. In an automated surveillance system, at each goal station, the robot stops and analyses the scene searching for victims, scene variations or abandoned/removed objects. *NOTE:* patrolling is a persistent task, it never ends, no notion of completeness.
- **Online coverage** (aka **exporation**): *cover* (with sensory perceptions) an unknown environment in the most "efficient" way. See below *Exploration Requirements*.

Vlada: Not feasible during Y3, let's focus on Patrolling
Luigi: PSE and PLDE had the priority

- **Sensor network deployment:** deploy robot sensors in order to completely cover a given region and guarantee uniform spatial density. This requires the deployment of a significant number of robots.

Vlada: Nice to have, but again, patrolling makes more sense with our resources
Luigi: here the list just aims at identifying interesting tasks (at least in principle)

In this context, **efficiency** can be evaluated by defining suitable metrics which take into account task completion time, traveled path, used communication bandwidth, etc.

A multi-robot task can be defined within one or multiple sorties.

Single-sortie Missions

In a single-sortie mission the two UGVs can be required to perform one of the following tasks:

- **Y3: Autonomous patrolling:** a map of the environment is already available and provided to the robots; robot autonomously plan their paths in order to visit/cover all the regions of the environment with equal frequency.
- **Y3: Assigned waypoints patrolling:** a map of the environment is already available and provided to the robots; users assign a set of waypoints to the robot team by using a suitable GUI; the robots negotiate the waypoints and then autonomously plan their paths in order to cyclically visit all the waypoints without interfering with each other.

Vlada: From the implementation point of view, isn't this actually an implementation of the previous point? At least the "cover" requirement? I would suggest to stop here and postpone the following point to Y4
*Luigi: the path-planning (input: start,goal,pcl -> output: safe trajectory) and trajectory-control modules can be arranged to be the same. The **cooperative** and **autonomous** planning algorithms which determine where to go (which waypoints) and when, are an important/required add-on.*

- **Y4: Exploration:** the UGVs autonomously explore and map the facility.
- **Y4: Coverage:** a map of the environment is already available and provided to the robots; the UGVs autonomously plan their paths in order to cover the environment with sensory perceptions.
- **Y5: Lost robot:** a UGV is missed and no connection to the base station. But the taken path is available from waypoint navigation. The second UGV and/or UAV are looking for it.

Multi-sortie Missions

In a multi-sortie mission the two UGVs can be required to perform one of the following tasks:

- **Parallel patrolling w/o collaboration:** Map is given, user defines areas by polygons or whatever to patrol, each robots patrols a single area.
- **Exploration + Patrolling:** in a first sortie, the robots explore the environment and build a map; in a second sortie the robots execute a patrolling task (autonomous or through assigned waypoints) by exploiting the previously built map
- **Exploration + Coverage:** in a first sortie, the robots explore the environment and build a map; in a second sortie the robots execute a coverage task by exploiting the previously built map

Target Implementations

Here we specify the multi-robot tasks that will be actually implemented for reviews and demos.

1. Pseudo-Collaborative Patrolling in a Static Environment (PCPSE)

This is a Vlada's proposal for TJEX/TEVAL 2016, Luigi is editing/commenting as well. Since Abel and Renaud expect to have multi-robot-mapping/cooperation not sooner than end of Y3, let's define lighter version of patrolling. We can still claim that it is multi-robot, but these robots will perform patrolling of distinct areas in parallel. Let's assume a patrolling mission in a static environment. A map of the environment is already available and was built in a previous

dedicated sortie. If we get this running during summer, we can build and test based on it until review, where additional inter-robot communication and planning could be added and tested.

See the dedicated page: [PCPSE](#)

2. Patrolling in a Static Environment ([PSE](#))

We can start experimenting with the simplest multi-robot task: a patrolling mission in a static environment (see *Issues->Dynamic Environments* below). A map of the environment is already available and was built in a previous dedicated sortie. In order to implement a PSE algorithm the following requirements are fundamental.

PCPSE - Table of Requirements

ID	Description	Status	Estimated Delivery Date	Key Partners	Notes	Issues/Questions
PSE1	Environment is static	-	-	-	-	-
PSE2	The environment map is already available and given as input to the robots	OK	Now available	ETH	-	-
PSE3	The robots are able to continuously localize themselves w.r.t. the given map	?	?	ETH	This means the robots share a single "map" reference frame	-
PSE4	Robots are able to continuously communicate and exchange their positions and relevant data	?	?	FRA, ROMA, ...	In principle, if a map is available and the environment is static, all the robot trajectories can be planned and shared beforehand (when mission starts) without requiring further data communications. Nevertheless, exact temporal execution of the planned trajectories cannot be guaranteed in a USAR environment and each robot can be seen as an high-dynamic obstacle by the other robots. Therefore, for robustness reasons, it is desirable that robots keep on continuously exchange information (at least their estimated positions) and possible plan updates.	See below <i>General Issues->Multi-robot communication</i>
PSE5	GUI definition/implementation for waypoints assignment	?	?	FRA, ROMA, CTU, ...	Patrolling can be completely autonomous (if the waypoints are autonomously planned by the algorithm) or end-users can explicitly assign a set of interesting goal stations. See more details in <i>Low-Level Mission/Task Definition</i> above.	-
PSE6	TRADR DB management for persistency	?	?	FRA, ROMA, ETH, CTU, ...	We need to define and store relevant data structures in MongoDB (e.g. maps, traversability, related infos). We need to implement the interface towards MongoDB.	-
PSE7	PSE algorithm implementation	WIP	T-Eval	ROMA	Currently testing/developing under simulation	...

PSE8	Dynamic mapping (at least considering robot occlusions)	TODO	T-Eval	ETH, CTU, ROMA	Mapping has to be able to clean robot ghost/trails from the built map.	Consider a starting narrow corridor, robot A starts behind robot B. Robot A will perceive robot B as an obstacle. Robot A mapper module must be able to clean the ghost/trail of robot B from the map, otherwise the path planner of robot A won't be able to compute a plan and move from its starting position. So we have to avoid robot A from being trapped by ghosts :-). In order to cope with that, without a decent dynamic mapping , we have to (1) select a sufficiently large region where to the start (2) suitably arrange the initial robot positions (3) suitably procrastinate the start of robot A (4) guide the robots to patrol two distinct and well-separated regions in order to avoid further problematic meetings (5) do some tests...
PSE+

To be defined:

- Do robots start from the same point or from distinct points?

3. Patrolling in a Low-Dynamic Environment (PLDE)

As a second step we can consider an environment with low-dynamic objects (see *Issues->Dynamic Environments* below). Also in this case, a map of the environment is already available and was built in a previous dedicated sortie. In order to implement a PLDE algorithm the following requirements are fundamental.

PLDE - Table of Requirements

ID	Description	Status	Estimated Delivery Date	Key Partners	Notes	Issues/Questions
PLDE1	Environment has some low-dynamic features (e.g. some obstacles were moved w.r.t. previous mapping sortie)	-	-	-	-	See below <i>General Issues->Dynamic Environment</i>
PLDE2	The environment map is already available and given as input to the robots	OK	Now available	ETH	-	-
PLDE3	The robots are able to continuously localize themselves w.r.t. the given map	?	?	ETH	This means the robots share a single "map" reference frame	-
PLDE4	Robots are able to continuously communicate and exchange their positions and relevant data	?	?	FRA, ROMA, ...	-	See below <i>General Issues->Multi-robot communication</i>
PLDE5	GUI definition/implementation for waypoints assignment	?	?	FRA, ROMA, CTU, ...	Patrolling can be completely autonomous (if the waypoints are autonomously planned by the algorithm) or end-users can explicitly assign a set of interesting goal stations. See more details in <i>Low-Level Mission/Task Definition</i> above.	-
PLDE6	Mapping module communicates the necessary map updates to the planner module (as early as possible) taking into account moved/changed obstacles	?	?	ETH	For efficiency reasons, it is desirable the mapping module also communicates a geometrical description of the changed regions to the planner module (so as to allow efficient local updates of	

					traversability/segmentation data)	
PLDE7	TRADR DB management for persistency	?	?	FRA, ROMA, ETH, CTU, ...	We need to define and store relevant data structures in MongoDB (e.g. maps, traversability, related infos). We need to implement the interface towards MongoDB.	-
PLDE8	PLDE algorithm implementation	WIP	T-Eval	ROMA	Currently testing/developing under simulation	...
PLDE+

To be defined:

- Do robots start from the same point or from distinct points?

Other Tasks

...

Multi-Robot Communication

Here we specify the communication interfaces and the data which need to be exchanged by the robots for the tasks of interest.

Required data

- Pseudo-collaborative Patrolling Static Environment (PCPSE): each robot must broadcast
 1. its position w.r.t. the shared map
 2. required sensory data, at the moment available ones are: flammable gasses, smoke, victims. Planned: fire and hot-spot detection in thermo
 3. its ability to proceed with the assigned plan - if not able, broadcast an alarm informing about major change in the assigned area
- Patrolling Static Environment (PSE): each robot must multicast/broadcast
 1. its position w.r.t. the shared map
 2. its planned path to the next target position
 3. required sensory data, at the moment available ones are: flammable gasses, smoke, victims. Planned: fire and hot-spot detection in thermo (usually only one robot bears thermocam).
- Patrolling Low-Dynamic Environment (PLDE): each robot must multicast/broadcast
 1. its position w.r.t. the shared map
 2. its planned path to the next target position
 3. local map updates
 4. required sensory data
- Exploration: each robot must multicast/broadcast
 1. its map (global or local map updates)
 2. its position w.r.t. the shared map (assuming individual maps overlap and map merging/registration can be successfully applied)
 3. its planned path to the next target position
 4. required sensory data

Multi-Robot Database Management

Robot1 and Robot2 cooperatively explore the environment and need to store their data in the DB. The two following approaches can be envisioned.

Centralized multi-robot localization and map fusion

Each robot writes its individual map into the DB. During the mission a map-fusion-global-optimizer receives/reads the individual maps from the DB, globally register (additional loop closure detections?) and fuse them in a *new multi-robot map*. This new multi-robot map is then sent back to each robot. Each robot re-localizes in it, replaces its individual map with the new received/queried multi-robot map and starts extending it and sharing its position w.r.t. It. In this approach, the DB stores an individual map for each robot + a new resulting multi-robot map. A mechanism should be decided to manage further individual map updates (which should trigger somehow the map-fusion-global-optimizer to start a new job).

PROS: kind of "simpler" centralized approach.

CONS: robustness issues w.r.t. network disruption.

Decentralized multi-robot localization and map fusion

Each robot receives/queries the individual maps coming from the other robots and is responsible of independently fusing them with its own map. Each robot writes its map updates to the DB and share them with the other robots by using network communication (or DB?). In this approach, the DB stores a map for each robot. In principle, if robots forms disconnected subnetworks each individual map can be different from the others; on the other hand, if robot network graph is complete (each pair of robots can communicate) the individual maps should coincide up to measurement noise and assuming perfect global optimization (and no critical loop closure errors). Here, the DB could also be used just to store the "best" individual map that is built by the robots. But how?

PROS: robust w.r.t. network failures

CONS: more complex (obviously)

Questions

- In a multi-robot context, do we need to use the DB (as a common blackboard) for sharing intermediate results? See for instance the decentralized multi-robot DB management above: in principle, we could use the DB just for storing a final acquired world representation (e.g. select the best map and just store this).

See also the discussion here

📄 https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Database_Management_for_Mapping_and_Planning_20160609

General Issues

- **Dynamic Environment:** the environment can change substantially over time. This has to be explicitly managed by both the mapping and planning modules. Environment can have low-dynamic or high-dynamic features depending of the considered time-scale and actual rapidity of the changes. **Low dynamic** environments are in general composed of static and low-dynamic entities that can be moved or changed at any time, such as small obstacles, barriers and walls. In this case, a simple and widespread approach is to detect and treat them as outliers and locally update the map. On the other hand, **high-dynamic** objects continuously change the environment and must be represented and tracked explicitly with suitable techniques such as multi-target tracking. In this latter case, obstacle avoidance techniques (relying on fast data acquisitions) are also in order after global path planning guarantees the existence of a path toward a designated goal on the basis of the available built map. In principle, each robot can be also seen as a moving obstacle by the other robots: in this case data sharing and distributed/coordinated motion planning can efficiently manage inter-robot collision avoidance.

Vlada: Y3: Lets deal with dynamic environment by halting execution and informing end-user that something major has changed - goes well with patrolling task

- **Mutual localization:** in order to collaborate the robots must know their mutual positions in their individual maps. This can be achieved by using suitable techniques such as *map registration*, *map merging*, *place recognition* (appearance-based localization), etc. Map registration is obviously possible when a *minimal overlap* among individual maps occur so as to allow a consistent map merging. In order to ease this process we can make the robots start from the same point in order to guarantee an initial minimal overlap.

Vlada: Let's concentrate on this feature in Y4

- **Multi-robot communication:** a multi-robot task can be actually performed as long as **inter-robot communication** is working and information are shared.
 - *Ensuring communication constraints:* in order to communicate, robots may be required to jointly satisfy some geometrical constraints at all times (e.g. maintain mutual line-of-sight visibility and maintain their mutual distance below a certain maximum threshold) or may be called for a rendezvous after communication loss lasts more than a certain time interval. This is required to guarantee better cooperation/coordination: map/information integration should be done as early as possible, since the availability of a shared map/information greatly facilitates the coordination between robots.
 - *Robustness:* even if one robot fails catastrophically, others should take over its subtask. Fault-tolerant methods should be devised. What actually happens when the robot communication completely falls down?
 - *Careful network analysis and profiling* must be performed in order to ensure that the actual network bandwidth is sufficient to allow all the required data to be exchanged among robots.

Vlada: For Y3, I propose to check mutual distance only to avoid collisions.

Vlada: I think the following points might be addressed for Y4 if we manage to get the inter-robot communication and robot-database running during Y3

- **Multi-robot heterogeneous collaboration:** w.r.t. the tasks defined above, define how to efficiently/properly consider each robot specific characteristics in the role assignment and team deployment. Can we/end-users assign different labels to each environment region beforehand in order to suggest preferred task assignments or team deployments? (e.g. label regions where samples have to be analysed in order to prefer their assignment to UGVs equipped with robotic arm).
- **Interface definition:**
 - How a single operator can efficiently guide the two UGVs and assign them a cooperative task? An efficient and (hopefully) simple interface should be defined to this aim.
 - Is the operator allowed to interrupt a robot for assigning it a different task?
 - Which are the allowed control modes of the single robots during a multi-robot task execution? (e.g. Am I allowed to move the camera during a patrolling?)
- Clarify the **control modes** as well as the technologies to develop the interaction, the communication and the collaboration between robots and humans.

- Discuss problems related to the integration with the TRADR ontology, the low-level database (MongoDB) and the high-level database (Stardog)




Exploration Requirements

If an **exploration** has to be performed by a team of robots, the process requires:

- *Cooperation*: avoid inefficient actions
- *Coordination*: avoid conflicts/collisions
- *Decentralization*: required in order to guarantee robustness if one or more robots fail catastrophically or communication is lost
- *Integrated Exploration* (Active SLAM): an efficient exploration strategy should tightly integrate mapping, localization and planning tasks. When selecting a new action, the process must take into account
 - the energy/time/risk cost (*planning*)
 - the expected information gain (*mapping*)
 - the associated localization potential (*localization*)
- *Minimal map overlapping*: a minimal overlapping among individual maps is required during multi-robot exploration in order to consistently register and merge the maps (mutual-localization, cooperative-mapping, map merging)
- *Ensuring communication constraints*: see above *General Issues*->*Multi-robot communication*->*Ensuring communication constraints*
- *Robustness*: even if one robot fails catastrophically, others should take over its subtask

Useful References

- Task taxonomy:  <https://redmine.ciirc.cvut.cz/attachments/download/533/Task%20taxonomy.pdf>
- Scenarios:  <https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Scenarios>

 [patrolling3-ramp.png](#)  (571 KB)  Freda, Luigi, 12/08/2017 04:21 PM

V-REP Simulation

Installing V-REP

1. Download version 3.2.2 (tested) from here http://coppeliarobotics.com/files/V-REP_PRO_V3_2_2_64_Linux.tar.gz
You don't need to compile anything. Just extract the files in your V-REP installation folder and you are ready to execute the main launcher (vrep.sh) from there.
Note: V-REP 3.3 does not work. You'll have to use 3.2.2 or fix the UGV script for 3.3
1. Set the environment variable `VREP_ROOT_DIR`: add in your `.bashrc` the following line
`export VREP_ROOT_DIR=<here you put the absolute path of your V-REP installation folder (which contains the launcher vrep.sh)>`

Updating and Compiling the *tradr-simulation* Stack

1. Open a shell, get into your tradr workspace and update your working copy of the repo [git@gitlab.ciirc.cvut.cz:tradr/tradr-simulation](https://gitlab.ciirc.cvut.cz/tradr/tradr-simulation)

```
$ cd tradr_ws/src/tradr-simulation/  
$ git pull
```

2. Then compile the workspace with your favorite catkin tool

Installing the *vrep_ugv_plugin* Package

Once you have updated and compiled the tradr-simulation stack, you have to copy the lib `tradr_ws/devel/lib/libv_repExtRos.so` in the installation folder `VREP_ROOT_DIR` (NOTE: this lib enables V-REP to get and parse track velocity command messages)

Testing the *vrep_ugv_simulation* Package

In order to test the package, two procedures are possible

1. Launch script

- open a terminal, source the tradr workspaces and execute

```
$ roslaunch vrep_ugv_simulation vrep_ugv_simulation.launch
```

- **press the play button on V-REP**

2. Manual

- open a terminal and run `roscore`
- open another terminal and execute

```
$ cd $VREP_ROOT_DIR  
$ ./vrep.sh
```

V-REP will be launched: from its *File* menu open an environment file.ttt in the package folder `vrep_ugv_simulation/data`. Use one of the following files: `ugv1_rescue_grouzers.ttt` or `ugv1_rescue_stairs_grouzers.ttt`

- open another terminal and execute

```
$ roslaunch vrep_ugv_simulation vrep_ugv_teleop_keyboard.launch
```

- **press the play button on V-REP**
- **N.B.:** if you kill the `roscore` you need to restart V-REP again.

Once you completed one of the above procedures, you should have your simulator running and a small window with title *"UGV TeleOp"* should appear. Keep the focus on that window (click on it) and you will be able to move the UGV with the arrow keys. By using the keys 'A','S','D','W' you are also able to change the configuration of its flippers in order to climb stairs.

The keyboard settings can be modified by editing the python script `vrep_ugv_simulation/script/vrep_ugv_teleop_key`.

V-REP with the Path Planner

1. Update and compile the path planning stack as described on this page https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning
2. Launch the UGV simulation on V-REP: open a new terminal, source the tradr workspaces and run

```
$ roslaunch vrep_ugv_simulation vrep_ugv_simulation.launch
```

3. press the play button on V-REP

4. Run the path planner with its RVIZ interface:

- If you want to run the **single-waypoint** path planner, open a new terminal and run

```
$ roslaunch path_planner sim_main_path_planner_ugv1.launch
```

- Otherwise, if you want to run the **multi-waypoint** path planner, open a new terminal and run

```
$ roslaunch path_planner sim_main_queue_path_planner_ugv1.launch
```

On the following page you can find a concise description of how to use the RVIZ interface in order to feed new goals to the path planner https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning

V-REP with the Multi-Robot Mapping and Path Planning

1. As above, update and compile the path planning stack as described on this page https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning
2. Compile the laser_slam workspace as described here https://github.com/ethz-asl/laser_slam/wiki/How-to-build-laser_slam-packages (official installation guide) https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Laser_SLAM_Workspace_Installation (you need to use a separate installation of PCL 1.8, this is short guide on how to solve compilation and RVIZ issues)
3. Launch the multi-UGV simulation on V-REP: open a new terminal, source the tradr and laser_slam workspaces and run

```
$ roslaunch vrep_ugv_simulation vrep_ugv_simulation_mapping.launch
```

4. Run the multi-robot mapping nodes:

```
$ roscd path_planner/scripts  
$ ./sim_launcher_mapping_ugv1n2
```

5. press the play button on V-REP

6. On RVIZ: the path planning is enabled for both robots. In order to select the WPs for **ugv1** press **"m"** and then stick the WPs on ugv1-traversability cloud and start (as explained in https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning). In order to select the WPs for **ugv2** press **"l"** and then stick the WPs on ugv2-traversability cloud and start.

You will see different xterms opening (one for each main group of nodes) and reporting you the messages of the distinct nodes. Attached to this page, you can find the diagrams of the frames and of the nodes.

Brief description of the architecture:

- robot i has its own laser_mapper, which is responsible for the localization of the robot i with respect to its own map_i and odom_i frames
 - from V-REP we provide the tf server with the transformations from /map_i to /map (as agreed during the last meetings this is an important input we must provide to the real system at the beginning of the mission)
 - one octomap manager integrates/merges the two maps coming from the single robots in single global map: the result is a single volumetric map which is available w.r.t. /map frame
- Hence, at present time, we still have a /map frame :-)

V-REP with Multi-Robot Patrolling

1. Update and compile the path planning stack as described on this page https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning
2. Compile the laser_slam workspace as described here https://github.com/ethz-asl/laser_slam/wiki/How-to-build-laser_slam-packages (official installation guide) https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Laser_SLAM_Workspace_Installation (you need to use a separate installation of PCL 1.8, this is short guide on how to solve compilation and RVIZ issues)
3. Download and compile in your TRADR workspace the **repo**: tradr-multirobot **@branch**: patrolling_sim_devel (see the README of the repo for detailed instructions)
4. Open a new terminal and source the TRADR workspace along with the new tradr-multirobot workspace, then execute

```
roscd patrolling_sim/scripts  
./sim_launcher_ugv1n2_light
```

5. press the play button on V-REP

What is going to happen?

- a proper V-REP world will be automatically launched
- a pre-built volumetric map will be loaded in the robot map managers (for convenience, this map has been already built and pushed in the repo; you can build it by yourself and use it)
- a pre-built graph (which nodes to visit) will be loaded and used by the patrolling agents (for convenience, this graph has been already built and pushed in the repo)
- the robots will start patrolling the environment: given a graph-representation of the environment (nodes represent reachable and safe regions, edges represent traversable paths joining them), patrolling requires continuously visiting all the graph nodes so as to minimize the time lag between two visits (node idleness)
- see the README file of the repo tradr-multirobot for advanced instructions

Videos: see in this shared folder https://drive.google.com/drive/folders/0B1oevBvCpw_vTGtIOUcxMI9Banc

V-REP and the Path Planner with Multimaster_fkie

The following instructions explain how to work with a multi-master V-REP simulation:

- V-REP runs on one host along with its ROS master
- your nodes run on a second host along with a second ROS master
- the two masters discover each other and sync their info by using multimaster_fkie (http://wiki.ros.org/multimaster_fkie?distro=kinetic): the distributed nodes will behave like they were under a single ROS master.

Steps:

1. install the package multimaster_fkie on both hosts
2. be sure that the ROS_IP is set on both hosts; this can be done by adding to the .bashrc files the following lines

```
# ROS: set automatically ROS_IP  
export ROS_IP=`hostname -I | head -n1 | awk '{print $1}'`  
if [ "$ROS_IP" != "" ]; then  
    export ROS_IP="$ROS_IP"  
fi
```

3. on the first host:
 - open a new terminal and run: \$ roscore
 - open a new terminal and run: \$ roslaunch tf_remapper multimaster_discovery_and_synch.launch

1. on the second host:
 - o open a new terminal and run: `$ roscore`
 - o open a new terminal and run: `$ roslaunch tf_remapper multimaster_discovery_and_synch.launch`
 - o open a new terminal and run: `$ roslaunch vrep_ugv_simulation vrep_ugv_simulation.launch`

1. on the first host:
 - o open a new terminal and run: `$ roscd path_planner/scripts; ./sim_launcher_ugv1n2`

V-REP Patrolling, Path Planning and Mapping over NIMBRO Network

Introduction: in order to develop and easily test the multi-robot network framework, we have prepared a VREP robot simulator which emits the same topics of an actual robot. Two instances of this simulator are launched, each one running on a distinct computer (as if they were two distinct actual robots). Laser mapper, traversability and path planner are normally launched (as on the actual robots). One of this computer is also used as a "tradr core" (by using a distinct roscore; clearly this is not a limitation since an additional computer can be also used without problem). Nimbros relays are activated and fully work.

In this simulation framework, you need two computers (PC). Two instances of V-REP will be launched. Each instance of V-REP acts as an actual robot and will emit the same topics of an actual robot (the main ones). The simulated robot 'sim1-robot' and a minimal tradr-core interface will run on the first computer. The second robot 'sim2-robot' will run on the second computer.

1. Update and compile the path planning stack as described on this page
https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning
2. Update, compile and install the vrep_ugv_plugin package as explained above.
3. Compile the laser_slam workspace as described here
https://github.com/ethz-asl/laser_slam/wiki/How-to-build-laser_slam-packages (official installation guide)
https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Laser_SLAM_Workspace_Installation (you need to use a separate installation of PCL 1.8, this is short guide on how to solve compilation and RVIZ issues)
4. Now go through the following steps
 - setup the hostname in the configuration file of the robots 'sim1-robot' and 'sim2-robot' in the dir tradr-network/tradr_relay_conf/; open a new terminal
``$ roscd tradr_relay_conf/conf``
``$ gedit sim1-robot.xml &` <-- set the prefix=ugv1 and hostname=localhost`
``$ gedit sim2-robot.xml &` <-- set the prefix=ugv2 and set the hostname of your second PC`
 - remove the labels 'ugv1' and 'ugv2' from any other actual robot configuration file <name>-robot.xml
 - then run the following command
``$ roslaunch tradr_relay_conf generate_launchfiles.py``
 - you can accomplish the above first steps on your PC/laptop and then synch the files between your PC/laptop and the second PC (you can use rsync for instance)
 - in a first PC, launch the first robot 'ugv1'
 - 1) launch VREP
``$ roslaunch vrep_ugv_simulation vrep_ugv_simulation_robot_ugv1.launch``
 press the play button
 - 2) launch the robot nodes and the nimbros robot relay
``$ roscd path_planner/scripts``
``$./sim_launcher_nimbros_robot 1` <-- set the CORE_HOSTNAME and the first arguments in the script`
 here the usage is: `./sim_launcher_nimbros_robot <robot-id> <core-hostname>`
 - in a second PC, launch the second robot 'ugv2'
 - 1) launch VREP
``$ roslaunch vrep_ugv_simulation vrep_ugv_simulation_robot_ugv2.launch`` <-- NB use the file with suffix ugv2!
 press the play button
 - 2) launch the robot nodes and the nimbros robot relay
``$ roscd path_planner/scripts``
``$./sim_launcher_nimbros_robot 2` <-- set the CORE_HOSTNAME and the first arguments in the script`
 here the usage is: `./sim_launcher_nimbros_robot <robot-id> <core-hostname>`
 - in the first PC, launch the minimal tradr core relay
``$ roscd path_planner/scripts``
``$./sim_launcher_nimbros_core true`` <-- set the first arguments therein ->
 here the usage is: `./sim_launcher_nimbros_core <RUN_SIM>` where RUN_SIM=true/false specifies if we are running a simulation or not. With REAL robots you can simply launch
``$./sim_launcher_nimbros_core`` on SIMULATION you instead have to run ``$./sim_launcher_nimbros_core true``
 - now you will see a global RVIZ and the patrolling starting
 - in order to let the robots plan a path on the patrolling graph you have to command each robot to load the map; for each robot
 - 1) enable on RVIZ the /ugvi/slam_marker (Interactive marker)
 - 2) right-click over it and select the action "Reset octomap and load previous map"
 - building a patrolling graph: in order to have this feature enabled you have to set BUILD_PATROLLING_GRAPH_ON_START="true" on both the scripts "path_planner/scriptsscreen_launcher_ugvmulti_lasermapper_pathplanner" and "path_planner/scripts/sim_launcher_nimbros_core"; on RVIZ, load the map on ugv1 (this procedure won't work on ugv2); press 'm' on the keyboard and stick a set of waypoint on the traversability carpet then right-click on one of the waypoint marker and select the action "Patrolling - send task"; the patrolling graph will be saved and can be used on further patrolling missions by setting BUILD_PATROLLING_GRAPH_ON_START="false" on both the scripts; now the patrolling will automatically start
 - using a saved patrolling graph: in order to have this feature enabled you have to set BUILD_PATROLLING_GRAPH_ON_START="false" on both the scripts "path_planner/scripts/sim_launcher_nimbros_robot" and "path_planner/scripts/sim_launcher_nimbros_core"
 - pause/restart patrolling: on RVIZ press 'm' and then 'q' for pausing or 'w' for restarting
 - if you want just to test the multi-robot path planning, you can set the variable ENABLE_PATROLLING=0 in both the scripts sim_launcher_nimbros_robot and sim_launcher_nimbros_core.
 In this case, you can set the waypoints for the two robots by using the keys "m" and "l".
 In order to select the WPs for the robot sim1 press "m" and then stick the WPs on ugv1-traversability cloud and start (as explained in

[TRADR setup](#) »

Multi-Robot Path Planning and Patrolling

Required Git repos and their branches

- **repo:** tradr-loc-map-nav @(master) **package:** path_planner
https://gitlab.ciirc.cvut.cz/tradr/tradr-loc-map-nav/tree/master/path_planner
- **repo:** tradr-loc-map-nav @(master) **package:** trajectory_control
https://gitlab.ciirc.cvut.cz/tradr/tradr-loc-map-nav/tree/master/trajectory_control
- **repo:** tradr-msgs @(master) **package:** trajectory_control_msgs
<https://gitlab.ciirc.cvut.cz/tradr/tradr-msgs>

Please, install [Octomap](#) packages in order to compile the path_planner package.

You can find a concise documentation at https://gitlab.ciirc.cvut.cz/tradr/tradr-loc-map-nav/blob/master/path_planner/README.md.

Launch the Basic Modules Needed by the Path Planner

Launch the ugv_multi drivers and the new ETHZ laser mapper

- on robot:

```
roslaunch nifti_drivers_launchers ugv_multi.launch
```

- on robot, start the new [ETH mapping](#):

```
roslaunch nifti_mapping_launchers mapAndNav.launch
```

```
roslaunch laser_mapper main_laser_mapper_tf_remapped.launch
```

Launching the Single-Waypoint Path Planner

First launch the basic drivers and laser mappers as described above.

Then, to run the path planner and connect it to the output of the ETHZ laser mapper

- on robot, start the planner:

```
roslaunch path_planner main_path_planner.launch
```

- on your laptop, start our rviz interface:

```
roslaunch path_planner rviz_path_planning.launch
```

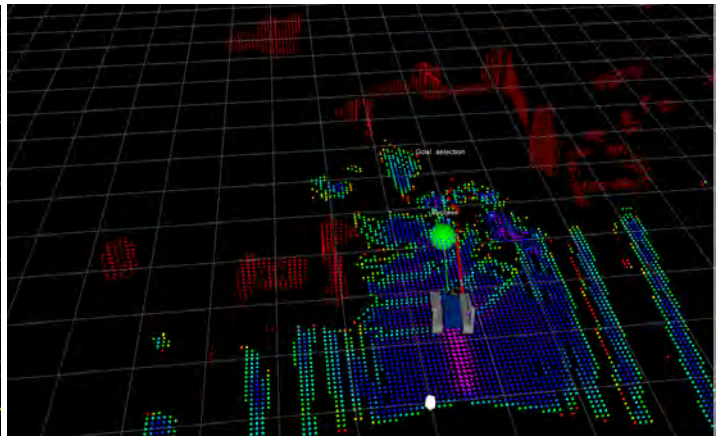
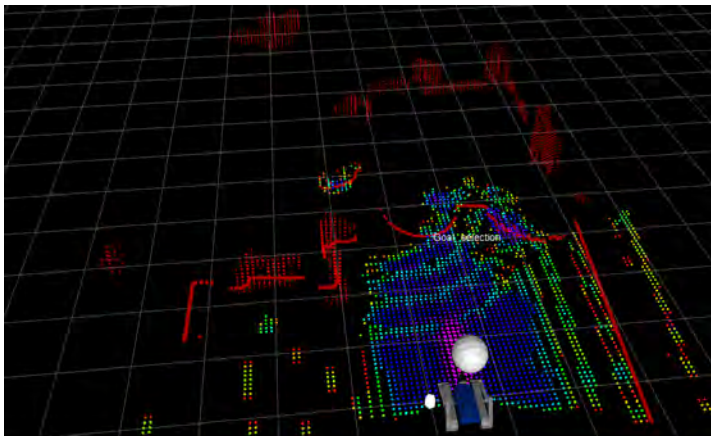
On RVIZ, you can set a goal for the planner by using the dedicated **interactive marker**. The interactive marker (a sphere) will appear over the robot when you launch the path_planner node.

1. Move the marker at your desired position (hold left click on it and move it w.r.t. image plane; if you also hold SHIFT button you will change the depth of the marker)
2. Then right-click on it and select from the menu the action "*Select Goal*". If you want to abort the goal once is selected, select the action "*Abort Goal*" from the same menu.
3. Text messages will appear over the marker explaining you what is happening. The **marker color** will change accordingly:
 - *Grey*, path planner is waiting for a goal selection
 - *Yellow*, the path planner is planning
 - *Green*, a path has been found
 - *Red*, the path planner could not find a path

N.B.: a path to the designated goal can be actually computed if the shown traversability map actually "connect" the goal and the robot positions. You may be required to wait few seconds till the traversability node actually complete the traversability map construction from the ICP mapper inputs.

Once a path has been found, the path planner will publish it towards the trajectory controller which will make the robot automatically follow the path.

There are **visualization topics** for the planned path, the normals of the point cloud and the merged point clouds. If you launch our script **rviz_path_planning.launch**, you will find everything already set up.



Launching the Multi-Waypoint Path Planner

First launch the basic drivers and laser mappers as described above.

Then, to run the multi-waypoint path planner and connect it to the output of the ETHZ mapper

- on robot, start the planner:

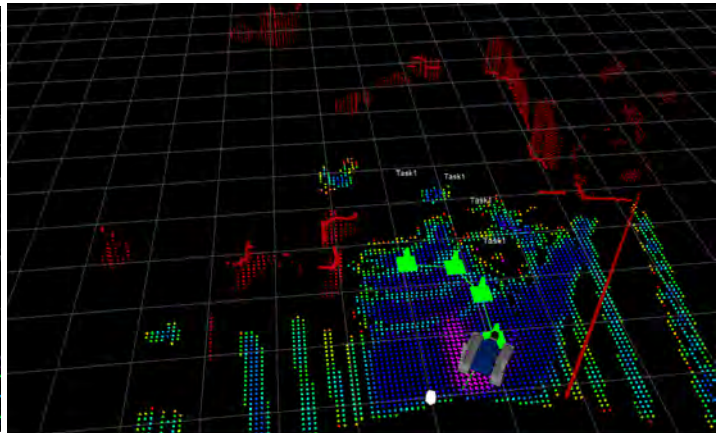
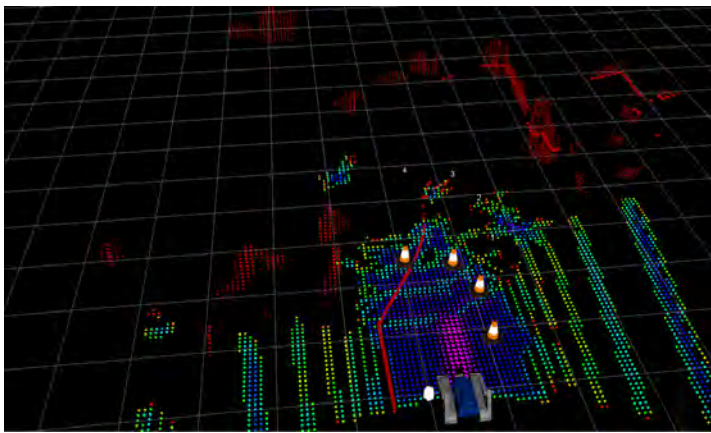
```
roslaunch path_planner main_queue_path_planner.launch
```

- on your laptop, start our rviz interface:

```
roslaunch path_planner rviz_path_planning.launch
```

On RVIZ:

- Press the key 'M' in order to add a new waypoint directly on the traversability cloud. You can move each created waypoint by holding right click on it and moving. The waypoints should automatically stick to the traversability cloud.
- Once you have selected your desired number of waypoints you can right click on one of them and select from the menu the action "Append Task". If you want the robot to continuously revisit the waypoints you set (**cyclic path**), then select the action "Append Cyclic Task".
- The **marker colors** will change accordingly:
 - *Orange*, the marker has not been added
 - *Yellow*, the path planner is planning
 - *Green*, a path has been found
 - *Red*, the path planner could not find a path
- Once the waypoints get green, you can right click on one of them and select from the menu the action "Stop the controller" in order to stop the trajectory control and the robot.



Multi-robot Patrolling, Path Planning and Mapping over Nimbro Network

Before starting this procedure, you have to build a map of the environment and save it.

In order to achieve that you can execute just the following step 1 and step 2 on one selected robot.

Once the selected robot is active and you get the global RVIZ interface running, move the robot around and build your map.

Once you are satisfied with the map, you can save it by enabling and using the /ugvi/slam_marker (Interactive marker): right-click on it and select the action "Save current map and trajectory".

(1) on your laptop (used as core) and on the two selected robots (for instance, you can use roma and eth robot):
setup the nimbro configuration file of each robot in the dir tradr-network/tradr_relay_conf/

- you can accomplish this first step on your laptop and then synch the files between your laptop and both the selected robots (you can use rsynch for instance)
- open a new terminal

```
`$ roscd tradr_relay_conf/conf`  
`$ gedit <your-robot-name>-robot.xml &`
```

see the README.md of the package tradr_relay_conf which explains how to setup the nimbro configuration file.
The labeling in roma-robot.xml and eth-robot.xml files should be the following
`roma-robot->prefix: ugv1` this corresponds to a multi-robot robot_id = 1
`eth-robot->prefix: ugv2` this corresponds to a multi-robot robot_id = 2
N.B. the label `ugv1` and `ugv2` are used to label the robots which will be actually used in the nimbro network for patrolling and path-planning.
These labels can be used for activating a pair of simulated robots OR a pair of actual robots, but only a pair of robots at once!
This means that when you use `ugv1` and `ugv2` for roma-robot::prefix and eth-robot::prefix, you should remove these labels from sim1-robot::prefix and sim2-robot::prefix, otherwise nimbro won't work.
- set and check the configuration variables in the two scripts "path_planner/scripts/screen_launcher_ugvmulti_lasermapper_pathplanner" and "path_planner/scripts/sim_launcher_nimbro_core", in particular set the same PATROLLING_MAP_NAME=DIAG3D (this corresponds to a folder patrolling_sim/maps/DIAG3D which must exist)
- then run the following command

```
`$ rosrn tradr_relay_conf generate_launchfiles.py`
```
- if you want you can now synch the files with rsynch between your laptop and both robots

(2) copy your map and graph files on each robot and on your laptop

- copy the saved map files ('map.bt map.ply trajectory.csv') on each robot and on your main laptop in the ".ros" folder
- to this aim, you can use for instance the script path_planning/scripts/copy_maps_to_robots

(3) on the first robot ugv1

- start the robot in the same position where you started the robot which built the saved map (reference starting position)
- kill all the nodes which are running on the robot: you can do that by executing the following command on the robot
`\$killall -9 screen; screen -wipe`
- open a new terminal
`\$ rosrn tradr_relay_conf generate_launchfiles.py`
`\$ roscd path_planner/scripts`
- then launch the main script
`\$./screen_launcher_ugvmulti_lasermapper_pathplanner <robot-prefix> <robot-id> <core-hostname>`
for instance on roma robot (tagged `ugv1`)
`\$./screen_launcher_ugvmulti_lasermapper_pathplanner 1 192.168.2.54`
and on eth robot (tagged `ugv2`)
`\$./screen_launcher_ugvmulti_lasermapper_pathplanner 2 192.168.2.54`
in order to kill everything you can use
`\$ killall -9 screen; screen -wipe`

(4) on the laptop

- check the file /etc/hosts and if needed add your robot hostname-IPs
- open a new terminal and run
`\$ rosrn tradr_relay_conf generate_launchfiles.py`
`\$ roscd path_planner/scripts`
`\$./sim_launcher_nimbro_core <RUN_SIM>`
where RUN_SIM=true/false specifies if we are running a simulation or not.
With REAL robots you can simply launch
`\$./sim_launcher_nimbro_core`
on SIMULATION you instead have to run
`\$./sim_launcher_nimbro_core true`
- now you will see a global RVIZ
- load the map on the first robot: enable on RVIZ the /ugv1/slam_marker (Interactive marker), right-click over it and select the action "Reset octomap and load previous map"

(5) on the second robot ugv2

- start the robot in the same position where you started the robot which built the saved map (reference starting position)
- repeat the actions described on the step 3 above for the first robot
- load the map: enable on RVIZ the /ugv2/slam_marker (Interactive marker), right-click over it and select the action "Reset octomap and load previous map"

(6) on the laptop: building a patrolling graph

- in order to have this feature enabled you have to set BUILD_PATROLLING_GRAPH_ON_START="true" on both the scripts "path_planner/scriptsscreen_launcher_ugvmulti_lasermapper_pathplanner" and "path_planner/scripts/sim_launcher_nimbro_core"
- on RVIZ, load the map on ugv1 (this procedure won't work on ugv2)
- press 'm' on the keyboard and stick a set of waypoint on the traversability carpet
- then right-click on one of the waypoint marker and select the action "Patrolling - send task"
- the patrolling graph will be saved and can be then used on further patrolling missions by setting BUILD_PATROLLING_GRAPH_ON_START="false" on both the scripts
- now the patrolling will automatically start

(7) on the laptop: using a saved patrolling graph

- in order to have this feature enabled you have to set BUILD_PATROLLING_GRAPH_ON_START="false" on both the scripts "path_planner/scriptsscreen_launcher_ugvmulti_lasermapper_pathplanner" and "path_planner/scripts/sim_launcher_nimbro_core"
- copy the saved file patrolling_sim/maps/DIAG3D/DIAG3D.graph (which is on your laptop in the map folder specified by the var PATROLLING_MAP_NAME=DIAG3D, which corresponds to the folder patrolling_sim/maps/DIAG3D) in the same folder patrolling_sim/maps/DIAG3D/ on both robots

(optional) on the laptop: pause/restart patrolling

- on RVIZ press 'm' and then 'q' for pausing or 'w' for restarting the patrolling

(optional) on the laptop: aligning a robot current map with a saved map

- copy the saved map (map.bt, map.ply and trajectory.cvs) on both robots and on the laptop
- on RVIZ enable the interactive marker and select "Reset octomap and load previous" in order to load the saved map (which should be copied on the selected robot)
- then if you want to align by using the interactive marker
`\$ roscd path_planner/scripts`
`\$./sim_launcher_core_interactive_point_cloud_publishers`

If you want just to test the multi-robot path planning, you can set the variable ENABLE_PATROLLING=0 in both the scripts sim_launcher_nimbro_robot and sim_launcher_nimbro_core.

In this case, you can set the waypoints for the two robots by using the keys "m" and "l".

In order to select the WPs for the robot sim1 press "m" and then stick the WPs on ugv1-traversability cloud and start (as explained in https://redmine.ciirc.cvut.cz/projects/tradr/wiki/Launch_path_planning).

In order to select the WPs for the robot sim2 press "l" and then stick the WPs on ugv2-traversability cloud and start.

Playing with the Path Planner on V-REP

You can find the details on this page

https://redmine.ciirc.cvut.cz/projects/tradr/wiki/V-REP_Simulation

Videos

- TJex 2016 Mapping and Point Cloud Segmentation
https://www.youtube.com/watch?v=74yp_fHpDpY&feature=youtu.be