Coordinated Multi-Robot Exploration: Out of the Box Packages for ROS

Torsten Andre[1], Daniel Neuhold[1], and Christian Bettstetter[1,2]
[1] Institute of Networked and Embedded Systems, University of Klagenfurt, Austria
[2] Lakeside Labs GmbH, Klagenfurt, Austria
torsten.andre@aau.at

***Abstract*—We present and evaluate new ROS packages for coordinated multi-robot exploration, namely communication, global map construction, and exploration. The packages allow completely distributed control and do not rely on (but allow) central controllers. Their integration including application layer protocols allows out of the box installation and execution. The communication package enables reliable ad hoc communication allowing to exchange local maps between robots which are merged to a global map. Exploration uses the global map to spatially spread robots and decrease exploration time. The intention of the implementation is to offer basic functionality for coordinated multi-robot systems and to enable other research groups to experimentally work on multi-robot systems. The packages are tested in real-world experiments using Turtlebot and Pioneer robots. Further, we analyze their performance using simulations and verify their correct working.**

I. Introduction and Motivation

Multi-robot systems for exploration and mapping have received a lot of interest using theoretical and simulative approaches [1]. Within recent years the reliability and performance of single mobile robot systems have increased significantly. Affordable robots are offered allowing research institutions to acquire multiple robots to conduct experiments focusing on the organization of multi-robot teams [2], [3]. Additionally, several frameworks to develop robot software are available. One prominent framework is the Robot Operating System (ROS) [4] offering a wide range of controllers for various hardware platforms. However, the support of multi-robot systems is still limited. We aim to enable future research in the area of autonomous multi-robot systems by presenting implementations in ROS of basic functionalities for coordinated multi-robot systems. Further, we discuss the performance of the implemented ROS packages.

During exploration, robots navigate to frontiers [5] to extend the known area until the complete (reachable) environment is explored. Using multiple rather than standalone robots allows to parallelize exploration by having individual robots explore distinct areas. It has been shown that explicit coordination of such systems can improve system performance, e.g., decrease exploration time of unknown environments [6]. Explicit multi-robot coordination requires a number of components. A map common to all robots is required to allow common understanding of the coordination. This map is constructed by exchanging local maps created by each robot. We use the terms "local" and "global" to distinguish contexts of a single robot and the complete multi-robot system. For example, a local map is the map created by each individual robot, while the global map includes local maps of all robots. Communication enables the exchange of data, e.g., local maps. Lastly, coordinated exploration utilizes communication and the global map to organize the multi-robot system by assigning frontiers to robots. We focus on self-organizing, decentralized systems where individual robots behave autonomously and do not depend on any additional infrastructure.

Our contribution is to present ROS packages that enable multi-robot exploration implementing the aforementioned required components, namely

- ad hoc communication between robots,
- construction of global maps from local maps, and
- exploration of unknown environments.

The remainder of this paper describes the capabilities of each package, implementation details, and evaluation. These packages are intended to be used by other researchers. Documentation and presentation of basic concepts shall ease their usage. To the best of our knowledge no integrated packages enabling coordinated multi-robot exploration are currently available on ROS. The presented ROS packages are available at:

- http://wiki.ros.org/adhoc_communication
- http://wiki.ros.org/map_merger
- http://wiki.ros.org/explorer

II. Related Work

A number of ROS packages have been published to allow communication and exploration. We briefly discuss their strengths and weaknesses.

A. Wireless Ad Hoc Communication

The majority of ROS packages implementing inter-robot communication in ROS rely on existing infrastructure in form of WLAN access points to which robots connect to. Communication packages focus on communication with short delays using UDP at the cost of decreased reliability. Currently available packages differ with respect to their functionalities and organization of communicated data. The *rocon_multimaster*[1] package implements building blocks that can be used by other packages to enable inter-robot communication, but does not offer communication itself. The *multimaster_fkie*[2] package allows the discovery of robots as well as unicast and multicast

[1]http://wiki.ros.org/rocon_multimaster/
[2]http://wiki.ros.org/multimaster_fkie

transmissions based on UDP. The *socrob_multimaster*[3] handles packet medium access using a TDMA scheme within the robot network. The Real-Time Database (RTDB) middleware [7] follows the concept of distributed shared memory where local databases are replicated to other robots allowing local, non-blocking access.

Existing infrastructure may not be assumed for all applications. For example, in search and rescue missions in devastated areas, robots should form an ad hoc network to setup their own communication infrastructure. The *cg_mrslam* package sets up an ad hoc network though does not seem to implement any routing protocols or diversity transmissions to increase reliability [3]. UDP is used on the transport layer.

In comparison, our ROS package implements Ad hoc On-Demand Distance Vector (AODV) for unicast and Multicast AODV (MAODV) for unicast multicast transmission. It uses automatic repeat request (ARQ) on data link and transport layers for unicast and multicast allowing reliable transmissions. The advantage is that applications using the communication package do not have to ensure reliability themselves. Developers may focus on their application and do not need to implement retransmission protocols themselves.

B. Map Merger

Distributed simultaneous localization and mapping (SLAM) has been discussed in the literature [8], [9], [10], though only few implementations are available on ROS. The *cg_mrslam* package implements distributed SLAM using condensed graphs while the *mapstitch* package allows to compute the overlap between maps and combines them accordingly. It was designed for offline usage. Our package builds upon *mapstitch* and extends it by application layer protocols allowing its usage during exploration (online usage).

C. Exploration

Early versions of ROS offered the *explore* and *exploration* packages but these are no longer maintained. Exploration packages for current ROS versions include *nav2d*[4], *frontier_exploration*[5], and *hector_exploration_node*[6], which implement single robot exploration. They identify frontiers [5] as navigation goals for a robot. In comparison, our package allows distributed coordination of multiple robots aiming to prevent robots from exploring identical areas. The authors of [1] give an overview of frontier selection and coordination methods.

The advantage of our three packages compared to all aforementioned ROS packages is the tight integration including application layer protocols. We offer out of the box coordinated multi-robot exploration and invite other researchers to further improve the implementations. The following sections discuss the three packages.

[3]http://wiki.ros.org/socrob_multicast
[4]http://wiki.ros.org/nav2d
[5]http://wiki.ros.org/frontier_exploration
[6]http://wiki.ros.org/hector_exploration_node

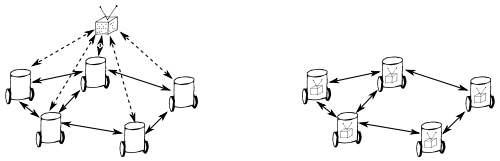


Fig. 1: (a) Central master manages interprocess communication (b) Each robot runs its own master managing local communication. Global communication is handled by a special ROS package.

III. Wireless Ad Hoc Communication

ROS was designed allowing primarily local interprocess communication managed by a master using the publish/subscribe communication pattern. The master matches publishers and subscribers based on ROS topics, i.e., communication channels between processes. To implement a multi-robot system, in a first step, all robots would have to connect wirelessly to a single master, illustrated in Fig. 1a. Only the master allows to establish direct communication channels between processes. Two processes need to communicate with the master to detect each other. If connections fail, processes rely on the master to reconnect them again. The solution having a single master per system has the drawback that, firstly, local interprocess communication is managed by an externally running master, i.e., two processes running on the same robot have to detect each other through the external master. Secondly, a single master decreases the reliability of the system as single point of failure.

Fig. 1b presents a distributed approach in which every robot executes its own master. Local interprocess communication is handled by the local master while global communication between robots is handled by a special communication package. Robots may communicate directly forming an ad hoc wireless network and do not rely on a central controller.

Ad hoc networks are a class of networks that do not rely on preexisting infrastructure. All robots participate in routing using routes that are adapted or created anew when the topology of the underlying network changes. The high degree of flexibility of these networks makes them ideal for mobile multi-robot systems. A well known ad hoc routing protocol is AODV [11]. Communication routes between robots are created on demand. If a route fails, i.e., the connection between source and destination is lost, the route will be repaired or a new route will be established. AODV is designed for mobile networks such as indoor robot systems.

We present the Ad Hoc Communication package, whose implementation strongly resembles AODV. In the following we sketch the integration of the Ad Hoc Communication package into ROS.

A. Features

Features of the Ad Hoc Communication package are as follows:

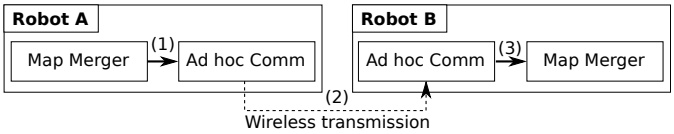


Fig. 2: Integration of the Ad Hoc Communication package in ROS.

- **Routing** allows robots to communicate via multiple hops to other robots which are not in the immediate neighborhood.
- **Multicast** allows to transmit from one robot to multiple robots, especially useful for dissemination of map data, for example.
- **ARQ** stands for automatic repeat request. If a frame is not acknowledged within a specified time, the sender will automatically repeat the transmission. The Ad Hoc Communication package supports both hop-by-hop ARQ and end-to-end ARQ.
- **Segmentation** allows to split data packets into multiple frames of smaller size. The IEEE 802.11a/g/n MAC layer supports payloads up to 2304 octets [12]. At the receiver side the frames are ordered and combined.
- **Ordering** ensures that frames and packets are delivered in the correct order.

B. Implementation and Integration into ROS

Fig. 2 illustrates the integration into ROS. In this example, robots A and B do not communicate directly but utilize the Ad Hoc Communication package using a service call (1). The service call includes destination, data to be transmitted, and a topic on which to publish the data at the destination. The Ad Hoc Communication package wraps the data into an extended MAC frame and transmits the frame using a raw socket (2). Using raw sockets allows the implementation of AODV and MAODV and easy modifications for future improvements running in user space and not in kernel space. Once robot B has received the frame successfully, the Ad Hoc Communication package publishes the received data under the topic specified within the packet (3). This implementation is flexible with respect to communication partners and completely transparent to ROS. Furthermore, this approach follows the ROS principle of communication by subscribing to topics.

IV. Map Merger

The task of the Map Merger package is to collect local maps of the robots and to merge them into a global map, which is used by the robots to navigate, explore, and coordinate. The mapping itself is performed only on the local map to decrease impact of errors in the global map. The Map Merger package may be distributed or centralized. If run centrally, each Ad Hoc Communication package will have to send its robot's local map to the central instance where the maps are merged. Having been merged, the global map needs to be disseminated to the robots. If distributed, robots share their local maps. The Map Merger package merges the local maps and delivers the resulting global map only locally, i.e., to the robot it is executed on.

The Map Merger package is inspired by the *map_stitch* package, but extents and improves it in numerous ways. Our package allows to be executed online during exploration and implements application layer protocols to exchange maps between robots.

A. Merging Process

Birk *et al.* describe the merging process conceptually and mathematically in [13]. The Map Merger package computes the transformation between two local maps M_1 and M_2 constructing the global map by joining the local maps maximizing the match over overlapping areas.

The transformation between coordinate systems is computed with OpenCV. By converting the ROS occupancy grid to a bitmap file, OpenCV's `estimateRigitTransform` method is used. It tries to match patterns within the local maps to determine the transformation. The decision to use OpenCV instead of more sophisticated models based on Bayesian updates [14] or filters [15] was made in favor of its easy implementation. A drawback is that overlapping local maps are required. However, in exploration missions it is meaningful to assume robots to start at the same position, e.g., the entrance of a building, so that maps overlap from the beginning on.

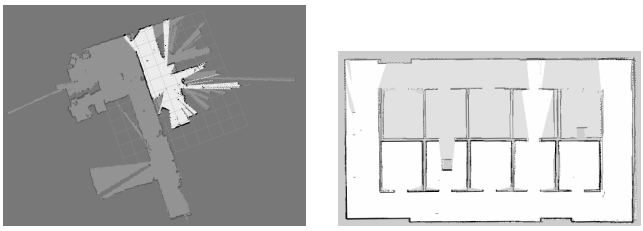
B. Features

Features of the Map Merger package are:

- **Map updates** are triggered if changes in local maps are detected.
- **New robots** are added and maps are exchanged automatically if the Ad Hoc Communication package reports a new robot in the system.
- **Robot positions** are transmitted in regular intervals. The package automatically converts other robots' positions to the correct coordinate system.
- **Integration Ad Hoc Communication package** allows the wireless exchange of maps between robots right out of the box. All topics are preconfigured.

V. Exploration

Robots may be utilized for a variety of scenarios, including search and rescue missions. For a complete search, robots need to ensure that they explore the entire environment. This requires robots to create a map to distinguish between already explored and unexplored space. The exploration package has the task to, firstly, identify frontiers [5] based on the current map. Secondly, frontiers are selected to be explored by the robot to extend explored areas. Exploration will finish if no more frontiers are left. Thirdly, in case of multi-robot system the package shall explicitly coordinate the robots to decrease exploration time [6].



(a) (b)

Fig. 3: Global maps created by (a) two Pioneer 3-DX robots with laser range scanners in a real world environment and (b) by four robots in a simulation each equipped with a laser range scanner. The light areas are the local maps overlayed on the global map for better illustration.

A. The Exploration Process

The exploration process is frontier based, i.e., transitions between explored and unknown space are detected and defined as navigation goals for a robot. Goals are ranked according to the Euclidean distance between the current position of a robot and a goal. Using the actual travel distance to frontiers would yield better results but has the drawback of consuming more resources and being error prone with currently available ROS path planners. Frontiers are clustered and coordinated based on auctioning [16] yielding good results compared to other methods [1]. The assignment of robots to clusters is performed based on the Hungarian method [17].

B. Features

Features of the Explorer package are:

- **Integration Ad Hoc Communication package** and
- **Integration Map Merger package** allow out of the box deployment. All topics and settings are pre-configured.
- **Coordination exploration** including frontier identification and coordinated assignment as described above.
- **Bid interpolation** aims to interpolates bids of other robots which did not send their bids for an auctioned cluster.

VI. Experiments

The packages are tested in real-world experiments using Turtlebot and Pioneer P3-DX robots in office corridors, labs, and partly in environments built for demonstration. All packages show reliable behavior, especially on the Pioneer robots. The Map Merger package benefits from the precise mapping of the Pioneer robots using laser range scanners. The Turtlebot is equipped with a Kinect in which map errors occur more often compared to the laser range scanner. Fig. 3a illustrates merged maps in our labs. For the Ad Hoc Communication package, we additionally executed long running tests on laptops to test its stability. Due to space constraints we refrain from publishing performance measurements for the Ad Hoc Communication package. AODV has been analyzed thoroughly in [18].

TABLE I: Errors e for resulting global maps at the end of explorations compared to a reference map.

R	0.05	mean	0.95
1	-0.24	0.11	0.68
2	-1.73	0.19	1.53
3	-1.32	1.83	6.20
4	0.74	6.40	14.67

VII. Simulations

We use simulations to quantitatively determine the performances of the multi-robot system, verify the correct implementation of the packages, and determine possible improvements for future work. Simulations are performed using the ROS stage simulator. Robot teams consist of $R = 1, 2, 3, 4$ robots. Simulations are performed $N = 40$ times for each robot team. Fig. 3b depicts a global map of the simulation environment. The map has dimensions of $26\,\mathrm{m}$ by $15\,\mathrm{m}$. Rooms have dimensions of $4\,\mathrm{m}$ by $4.9\,\mathrm{m}$. Robots start in the upper left corner of the map facing to the right. To solely determine the performance of the exploration package, we use error free communication.

A. Map Merger

We evaluate the quality of the global maps (see Fig. 3). To quantitatively determine mapping errors, we compare the number of explored pixels in the resulting global map to a reference map. The reference map is created by a single robot exploration. This accounts for the high impact the quality of the local maps have on map merging. We compare the global maps created by each robot individually at the end of explorations.

Maps consist of $P = P^o + P^e$ pixels of which P^o pixels indicate obstacles and P^e explored space. We approximate the error $e_n = (\frac{P_n^e}{\widehat{P^e}} - 1) \cdot 100$ for a simulation $n = 1, \ldots, N$, where $\widehat{P^e}$ is the reference value derived from the reference map. If the global map and reference map are identical, the number of explored pixels P_n^e and the reference value $\widehat{P^e}$ will have the same value, i.e., $e_n = 0$. Errors $e < 0$ result from local maps slightly shifted with respect to each other increasing the width of obstacles and thus reducing explored space. Errors $e > 0$ occur when local maps are slightly rotated or shifted in such a way that the map is invalidly extended in one or more directions.

The method for comparison is chosen because other techniques such as feature extraction, edge and line detection did not yield reliable results. While this metric is vulnerable to errors, comparing the quantitative error qualitatively to maps, the metric yields reliable results.

Tab. I shows the arithmetic mean of e and 0.05 and 0.95-quantiles over all N simulations. For up to three robots the maps are merged very precisely ($e \approx 0$) with small variance. For $R = 4$ robots, the merging process fails more often tending to increase the size of the global map due to misaligned mergers. We continue to evaluate the impact of map errors when using multiple robots on the Explorer package.

B. Explorer

We use the same simulation as described above. For each run $n = 1, \ldots, N$, robot $r \in 1, \ldots, R$ takes time T_n^r to finish its exploration while traveling distance d_n^r. The exploration for a robot r finishes when no more frontiers are available. The exploration of the environment is completed at time T_n once all robots have completed exploration, i.e., $T_n = \max(T_n^r)$. This is a lower bound for the exploration time. The global map may be available in advance.

TABLE II: (a) Cumulative travel distance for multi-robot team of size R in meters. (b) Team exploration time $\bar{T}$ in seconds.

(a)

R	0.05	mean	0.95
1	65	81	96
2	93	121	157
3	127	154	176
4	196	274	371

(b)

R	0.05	mean	0.95
1	865	945	1001
2	649	783	942
3	673	776	893
4	847	1208	1673

Tab. IIa shows the mean travel distance per robot team $\bar{d} = \frac{1}{N}\sum_{n=1}^{N}\sum_{r=1}^{R} d_n^r$ in meters and 0.05 and 0.95-percentiles. With an increasing number of robots, the cumulative travel distance increases as expected. Multiple robots have to travel the same paths to reach their goals. In comparison, the mean travel distance per robot decreases for the multi-robot systems compared to a single robot. A single robot treks $81\,\text{m}$, while three robots travel $\frac{154\,\text{m}}{3} \approx 51\,\text{m}$. The comparably large range between the 0.05 and 0.95-quantiles is due to the path planning. With different timings of received sensor data between simulations, robots may plan paths to identical destination points differently. For four robots the mean travel distance increases to $69\,\text{m}$ suggesting a decrease in exploration efficiency and likely longer exploration times.

Tab. IIb shows the mean exploration times $\bar{T} = \frac{1}{N}\sum_{n=1}^{N} T_n$. With up to $R = 3$ robots the exploration time decreases compared to $R = 1$ robots. For $R = 4$, as suggested by the travel distance, $\bar{T}$ increases due to increased map errors (see Tab. I) and robot interference. We consider the exploration progress for a more detailed analysis. Similar to the pixels for the Map Merger evaluation, robots use occupancy grids dividing maps into $M = O + E$ tiles out of which O tiles are obstacles and E tiles are explored. We track the exploration progress E_t at time t. A reference value $\widehat{E}$ is determined through complete explorations by a single robot. Again, we compute the mean number of explored elements $\bar{E}$ for all N simulations.

Fig. 4 shows the normalized mean number of explored elements $\frac{\bar{E}_t}{\widehat{E}}$. The vertical lines indicate the mean exploration times $\bar{T}$ according to Tab. IIb. With an increasing number of robots, the explored area E_t is extended faster by spatially distributing the robots decreasing exploration time. For example, at $t = 400$ one robot explores 0.66 of the area while three robots explore 0.82 in the same time. For $R = 4$ robots, however, the exploration time increases while Fig. 4 seems to indicate faster progress. The reason is increased map errors of

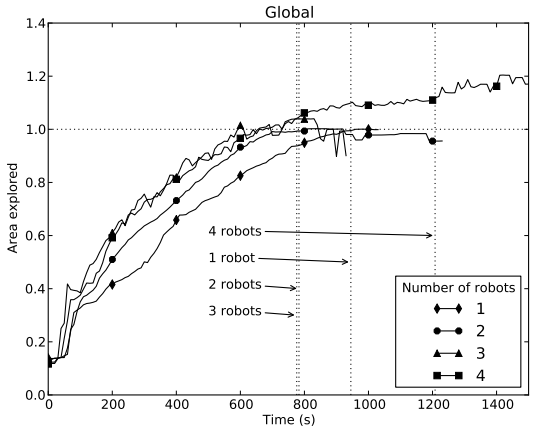


Fig. 4: Exploration progress by area on map depicted in Fig. 3b

the global map as depicted in Tab. I, invalidly increasing the size of the map. Therefore, the explored area may become larger than the actually existing area, i.e. $\frac{\bar{E}_t}{\widehat{E}} > 1$. Fig. 4 verifies that the robots spread through coordination, but also stresses the importance of reliable map merging.

So far we have determined the exploration time based on the last robot completing exploration. This includes a negative bias when the whole environment is explored, but one or more robots identify new pseudo frontiers in their own maps, i.e., frontiers that were only created due to map errors. Pseudo frontiers may be generated when obstacles in local maps erase each other due to alignment errors. This can be prevented by having obstacles never be deleted by explored space which may yield pseudo obstacles. While pseudo frontiers decrease efficiency, pseudo obstacles may keep robots from completing the exploration process. Pseudo frontiers prolong exploration because either a robot travels to a pseudo frontier detecting its mistake or such frontiers cannot be reached. Unreachable frontiers take additional time due to extended path planning. Ideally, robots complete exploration simultaneously. Though pseudo frontiers lead to a high variance of exploration times per robot T_n^r.

We show the corresponding cumulative density function (CDF) of T_n^r values in Fig. 5, i.e., the probability when individual robots finish exploration due to the absence of frontiers. The vertical lines again indicate the mean exploration time $\bar{T}$ for the corresponding team size. For example, while for multi-robot systems consisting of $r = 3$ robots $96\,\%$ of the robots finish exploration within $t = 900\,\text{s}$, for $r = 1$ only $21\,\%$ finish exploration in the given time. With an increasing number of up to three robots, the probability to finish exploration earlier increases supporting previous results of faster exploration and verifying the correct implementation of the packages. Only in case of $R = 4$ robots the probability decreases significantly having two reasons. The first reason is aforementioned pseudo frontiers. The map error e can be correlated to long exploration times. With increasing map error e for increasing number of robots, robots are more likely

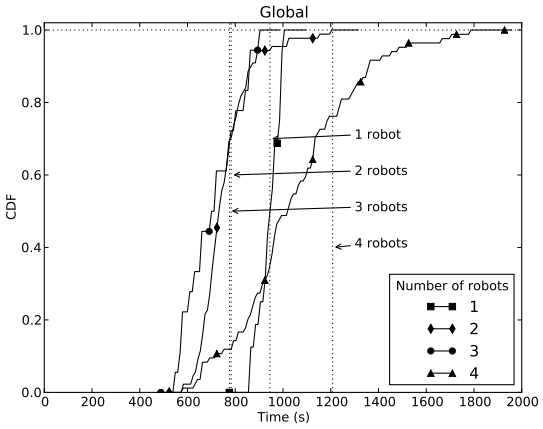


Fig. 5: Probability of exploration progress on map depicted in Fig. 3b

to take a long time to finish exploration. For example, for $r = 4$ robots $50\,\%$ of all robots take exploration times longer than $1021\,\mathrm{s}$. Secondly, increased robot interference decreases efficiency because the environment is too small and simple for four robots to efficiently distribute.

VIII. Conclusions

We presented three ROS packages enabling distributed, coordinated multi-robot exploration and invite other researchers to modify and extend according to their needs. Specifically, the Ad Hoc Communication package allows reliable unicast and multicast while not requiring additional infrastructure such as WLAN access points. It integrates transparently into the ROS ecosystem, allows transmission of arbitrary data, and supports basic functionalities of communications such as segmentation, in-order transmissions and acknowledgments. The Map Merger package allows to build a global map from multiple local maps and transform between coordinate systems. The Explorer package keeps track of already explored space and implements coordinated multi-robot exploration by assigning distinct frontiers to robots using auctioning and the Hungarian method. In comparison to other packages available on ROS, the packages include application layer protocols allowing out of the box execution and do not require central control. The packages are independent of the used hardware and have been tested on Turtlebot and Pioneer 3-DX robots. Additionally, simulations show the performance of the packages and verify their correct implementation. They also highlight the importance of accurate global maps having a direct impact on the exploration time. Extensions and modifications to the packages can be implemented easily according to the individual needs. We continue to improve and update the packages.

Acknowledgments

This work was partly supported by the ERDF, KWF, and BABEG under grant KWF-20214/24272/36084 (SINUS). It was performed in the research cluster Lakeside Labs. We thank Peter Kohout and Günther Cwioro for helping with the implementations.

References

[1] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, pp. 427–444, Nov. 2012.

[2] A. Marjovi and L. Marques, "Multi-robot topological exploration using olfactory cues," in *Distributed Autonomous Robotic Systems*, A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, L. E. Parker, and K. Sty, Eds. Springer, 2013, vol. 83, pp. 47–60.

[3] M. Lazaro, L. Paz, P. Pinies, J. Castellanos, and G. Grisetti, "Multi-robot slam using condensed measurements," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, Nov. 2013.

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA) - Workshop on Open Source Robotics*, May 2009.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Computational Intelligence in Robotics and Automation (CIRA)*, July 1997.

[6] T. Andre and C. Bettstetter, "Assessing the value of coordination in mobile robot exploration using a discrete-time Markov process," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nov. 2013.

[7] F. Santos, L. Almeida, P. Pedreiras, and L. Lopes, "A real-time distributed software infrastructure for cooperating mobile autonomous robots," in *Proc. Int. Conf. on Advanced Robotics (ICAR)*, June 2009.

[8] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using constrained factor graphs," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010.

[9] L. Carlone, M. Ng, J. Du, B. Bona, and M. Indri, "Rao-blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, May 2010.

[10] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Group mapping: A topological approach to map merging for multiple robots," *IEEE Robot. Automat. Mag.*, vol. 21, no. 2, pp. 60–72, June 2014.

[11] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), Internet Engineering Task Force, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[12] *IEEE Standard for Information technology - Specific Parts - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Computer Society Std., 2012.

[13] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384–1397, July 2006.

[14] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2003.

[15] N. E. Özkucur and H. L. Akın, "Cooperative multi-robot map merging using fast-SLAM," in *RoboCup 2009: Robot Soccer World Cup XIII*. Springer, Jan. 2010.

[16] R. Simmons, D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proc. AAAI National Conf. on Artificial Intelligence*, July 2000.

[17] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sept. 2008.

[18] C. Perkins, E. Royer, S. Das, and M. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Pers. Commun.*, vol. 8, no. 1, pp. 16–28, Feb. 2001.