Modelling a TurtleBot3 Based Delivery System for a Smart Hospital in Gazebo

Ruslan Safin
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan Federal University
Kazan, Russia
RuslanRuSafin@stud.kpfu.ru

Roman Lavrenov
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan Federal University
Kazan, Russia
lavrenov@it.kfu.ru

Kuo-Hsien Hsia
Department of Electrical Engineering
National Yunlin University of Science & Technology
Yunlin, Taiwan
khhsia@yuntech.edu.tw

Elena Maslak
Department of Industrial Engineering and Manufacturing
Volgograd State Medical University
Volgograd, Russia
eemaslak@yandex.ru

Natalia Schiefermeier-Mach
Health University of Applied Sciences
Innsbruck, Austria
natalia.schiefermeier-mach@fhg-tirol.ac.at

Evgeni Magid
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan Federal University
Kazan, Russia
magid@it.kfu.ru

2021 International Siberian Conference on Control and Communications (SIBCON) | 978-1-7281-8504-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/SIBCON50419.2021.9438875

***Abstract*—The design of a "smart hospital" environment is described in this paper. Mobile ground robots perform transportation tasks between multiple stations located in different rooms while navigating in an environment with moving objects such as humans and other mobile robots. The robot is equipped with a distance sensor (LIDAR), based on the indicators of which objects are detected and the robot is localized. Robots can be assigned tasks to be executed through a centralized interface. Tasks are assigned to a specific robot, and, depending on the type of task, the robot is autonomously directed to the station associated with the task in the building. We are considering the concept of defining possible robot behaviors as a finite set of states with certain transitions. To test the system, a hospital map was constructed in a Gazebo simulation.**

***Index Terms*—Hospital service robot, robot navigation, path planning, task-based robotic system, service robots**

I. Introduction

In robotics, in addition to the industrial field, noticeably developing the use of robots in the role of assistants, operating in the same environment with humans and other robots [1]. The development of this area is especially noticeable in relation to places where there are constant routine tasks of transporting objects from one point to another or being a social companion - such as hotels, hospitals and offices. The use of robots for transportation tasks would significantly reduce the cost of resources and work time of people for some routine tasks [2], however, this approach requires solving such problems as robust navigation, the safety of movement of robots, and the logistics of performing tasks by the robot. A variety of mobile robots design performing service tasks (mainly transportation) in conditions of close movement in a dynamic environment, dynamic obstacle avoidance approaches have been studied [3].

Solving these problems requires testing the robotic system for a variety of situations. The system described in the article was tested using virtual simulation, which, together with the use of software tools, made it possible to debug the behavior of robots in accordance with the criteria of the tasks at no significant cost. The virtual testing was carried out in a simulation of a floor of a medical facility, with personnel moving in rooms and corridors and robots performing delivery tasks in the same environment. The simulation allows the use of robot models that match the actual physical characteristics and are equipped with sensors. Tasks were assigned to the robots through the interface, after which the robots were sent to perform them (usual task is bypassing several "stations" located in different rooms of the simulated hospital) in accordance with the priority of the tasks, while avoiding collisions with other traffic participants, which required the use of dynamic obstacle avoidance algorithms in robot navigation setup. Robust navigation of mobile robots was implemented using the TEB planner - the path planning algorithm with underlying method called Timed Elastic Band was used [4],

[5]. The next section provides an overview of related work in order to highlight relevant work and proposals. Next, our approach and technical details are discussed (sections 3 and 4), followed by a description of the virtual test environment (section 5) and the simulation tests results (section 6).

II. Related Work

One of the most important tasks for operating a robot in an environment consisting of many corridors and rooms in which other moving objects (for example, people, other robots) also operate is correct navigation, which should not only ensure that the robot reaches its next destination, but also avoid collisions with other moving objects. In [3] authors have proposed a moving and static obstacle avoidance method for an omni-directional mobile robot with a platform for loads, using the MKR robot (Muratec Keio Robot) as an example. Their approach is based on path planning using virtual potential fields [6]. However, this approach is not suitable for use by differential drive robots, the navigation algorithms for which usually shows effectiveness and robustness with static obstacles, however, which do not always provide possibility for handling dynamic obstacle avoidance problem [7]. In another approach, authors propose a mobile robot system designed to solve the problems of delivering various medical items from one station to another in the hospital environment [8]. Nomadic XR4000 is used as a mobile platform. The hospital ceilings are equipped with florescent lamps, which are used by the robot as natural landmarks to determine its own position and orientation in the operating area. Next work considers the problem of coordinated execution of separate tasks by several robots, where a robot can incrementally receive tasks from different providers [9]. The authors described robots sparse- coordination what requires of robots keeping track of their states and being able to request states of each other's when needed. The approach is based on representing tasks as Instruction Graphs and sparse-coordination using Sparse-Coordination Instruction Graphs [10]. Another approach is the problem of task management for a mobile robot operating in an environment with people with the ability to receive new tasks from them [11]. Initially, robots receive tasks through a centralized source (web interface), and perform them without the ability to change the process of completing tasks during their execution. The authors presented a solution allowing task interruption, as cancelling tasks, querying the current task status of the robot, assigning a new task to be completed immediately or later, all of above using voice dialog interaction with the robot. The results were tested based on various scenarios for interrupting the current robot tasks.

III. Task-Based Architecture Implementation

A. Task receiving and execution

The workflow of the developed system can be described as follows. The user can use the GUI to assign a task to the robot by selecting the type of task, the station (location) where the task will be performed and where the robot should follow, the priority of the task, according to which the task sequence is built, the id of the robot to which the task will be assigned, and, optionally, task completion time - in order to define how long robot should wait when arrived at the station. If waiting time is not specified, the robot will start the next task (or switch to the WAIT_FOR_GOAL state) immediately after completing the current task. The states are described in more detail in the next subsection.

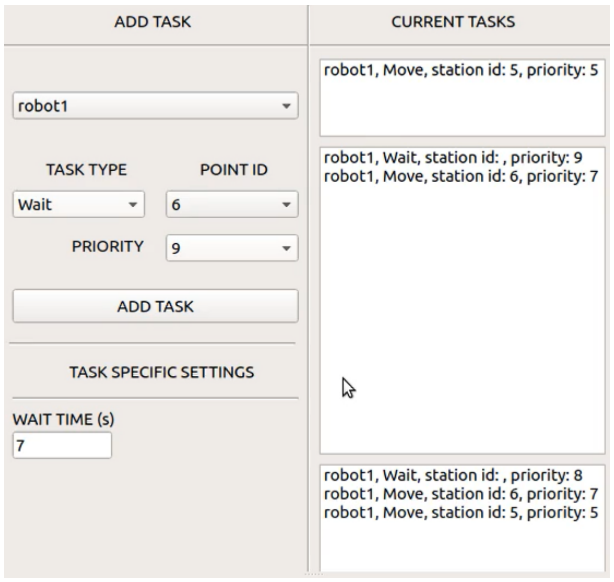
Task data structure is described in Table 1. After the task is assigned, the task manager sends it to the robot, which, on receiving new task, sorts its list of tasks according to their priorities and proceeds to perform the highest priority task from the list. Tasks can be assigned to a robot without waiting for the previous ones to be completed, and each robot has its own list of tasks, while the task manager stores all tasks and displays the current status of tasks in the GUI, dividing them into running, completed, and pending tasks. Task statuses are updated when robots transitions its state from one to another. For example, the robot notifies the task manager after completing a task or or when it starts executing the next task. The appearance of the interface is shown in Figure 1. At the moment, two basic types of tasks are implemented: waiting and autonomous movement to the station.

TABLE I
Task object fields description

Field	Description
id	Unique identifier
priority	An integer value that defines task execution sequence, from 0 to 10
isDone	Boolean value, sets to true when task is executed
isCurrent	Boolean value, sets to true when task is being started to execute
goalId	Unique identifier of the station on a map
taskType	Determines what the robot should do
robotName	Determines to which robot task is assigned
waitTime	Optional value, represents the time robot should wait after task is executed

B. Finite state machines with SMACH

The entire robot behaviour is described using a finite state machine with different types of transition from one state to another, depending on the conditions of completing the current state. The structure of finite state machines was implemented using the SMACH package [12], free and available for use in the ROS (Robot Operating System) framework. SMACH is a package providing task-level architecture for complex robot behavior, which allows to create state machines (or state containers), define their hierarchy using nested finite state machines, introspect states, state transitions and data flow between them in runtime, and so on. This approach allows to completely determine the behavior of the robot, dividing it into states, and manage states using conditional transitions and data transferring from state to state during the transition. At the moment, the structure visualization of the implemented



robot1, Move, station id: 5, priority: 5

robot1, Wait, station id: , priority: 9
robot1, Move, station id: 6, priority: 7

robot1, Wait, station id: , priority: 8
robot1, Move, station id: 6, priority: 7
robot1, Move, station id: 5, priority: 5

Fig. 1. Graphical user interface of task manager

state machine is shown in Figure 2. In Table 2 possible robot states and transitions between them are described.

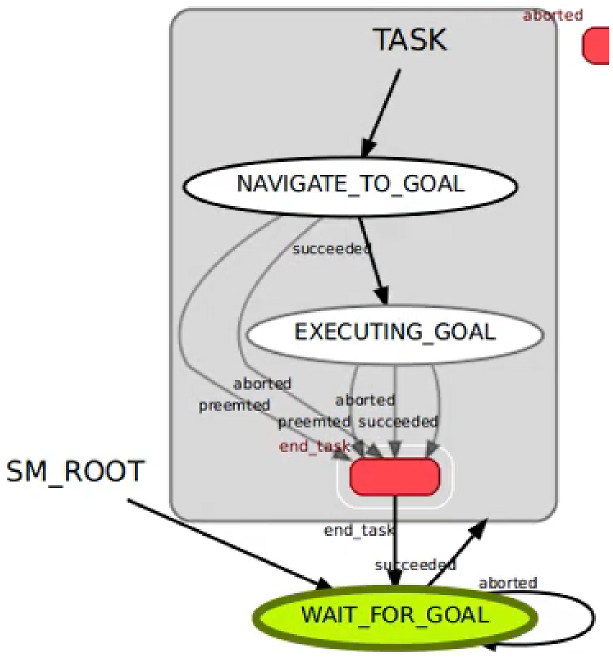


Fig. 2. Current state machine structure

The robot's behavior during the task execution is transferred to a nested state machine, the transition from which is currently possible with the status ”success”, which means that the task is completed, after which the robot is in the ”task waiting” state, which describes the process of waiting for a new task. If there are outstanding tasks in the robot's task list, the robot will immediately switch to the NAVIGATE_TO_GOAL state, thus starting to move to the station described in the task being performed.

State changing is always accompanied by a transition. Transitions from the NAVIGATE_TO_GOAL state are inherited from the move_base package [13], which is used as an interface and to control to the robot navigation stack. The transition ”success” is triggered if move_base reported the correct reaching of the navigation goal. ”Preempted” transition is executed if processing of the goal was canceled by receiving another goal. ”Aborted” transition is called if goal was terminated by the action server without requesting it, for example, if specified goal is unreachable or if the robot is stuck [14]. For now, this is sufficient for simple navigation tasks, but it is planned to add more states and transitions to describe more complex behavior.

TABLE II
ROBOT BEHAVIOUR STATES DESCRIPTION

State	Description
NAVIGATE_TO_GOAL	Robot autonomously navigates to the point specified in task
EXECUTING_GOAL	Robot executes goal (i.e. waits of task time is specified)
WAIT_FOR_GOAL	Robot stands still until goal is received

IV. Movement Rules Model Implementation

A. Navigation

As the robot's navigation module, there was used ROS integrated move_base package, which provides tools for route planning [15], [16], localization and SLAM [17], [18]. The move_base package provides an implementation of an navigation action module, which, given a goal in the map, will attempt to reach it with a mobile base [19]. During the operation this package combines the local and global planner functions to accomplish global navigation task [13]. As local path-planner TEB local planner was used. ROS teb_local_planner package implements an optimal local trajectory planner for navigation and control of mobile robots as a plugin for the ROS navigation package. The initial trajectory generated by a global planner is optimized in runtime with regard to minimizing the trajectory execution time, obstacle avoidance and compliance with constraints such as satisfying maximum velocities and accelerations. The optimal trajectory is efficiently obtained by solving a sparse scalarized multi-objective optimization problem. Weights can be provided as input parameters to the optimization problem in order to specify the behavior in case of conflicting objectives [4], [5], [20]. An extension of the algorithm was implemented for this package due to the tendency of local planners (such as TEB local planner) to get stuck on a locally optimal trajectory as they are unable to transit across obstacles. A subset of admissible trajectories of distinctive topologies is optimized in parallel. The local planner is able to switch to the current globally optimal trajectory among the candidate set.

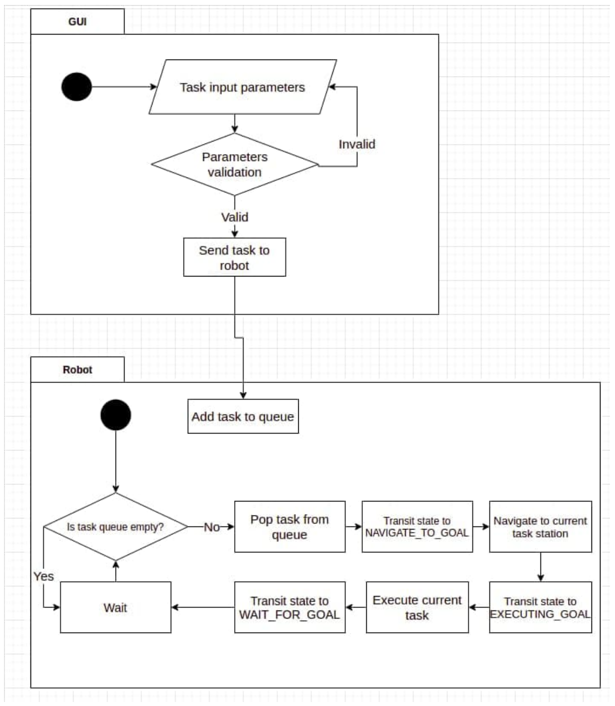


Fig. 3. Application block diagram

Distinctive topologies are obtained by utilizing the concept of homology/homotopy classes [21]–[23].

For robot localization purposes, the AMCL (Adaptive Monte-Carlo Localization) method is used, which provided by amcl ROS package. It implements the adaptive Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map [24], [25]. In a hospital environment, in addition to static obstacle avoidance, the robot needs to avoid collisions with moving objects such as hospital staff, patients and other robots performing their service tasks. For handling dynamic obstacle avoidance problem, the plugin called costmap_converter::CostmapToDynamicObstacles provided by the costmap_converter ROS package was used. The costmap_converter package provides plugins designed to convert points from costmap_2D (points represent data about the occupied space around the robot and are provided by some sensor, in our case, LiDAR was used) into geometric primitives. Primitives can be points, lines or polygons, and they represent obstacles on the map, and then in working with, in our case, navigation, all obstacle processing is performed with the resulting geometric shapes, abstracting from the previous set of individual points. These transformations are performed at runtime [27], [28]. Costmap_converter provides parameter to select a plugin to specify the algorithm by which the clusters will be determined. Below some of the possible algorithms are described. Plugin called CostmapToPolygonsDBSMCCH converts occupied cells to a set of convex polygons using the DBSCAN algorithm [29], this approach also was used in described system. This algorithm was chosen because convex

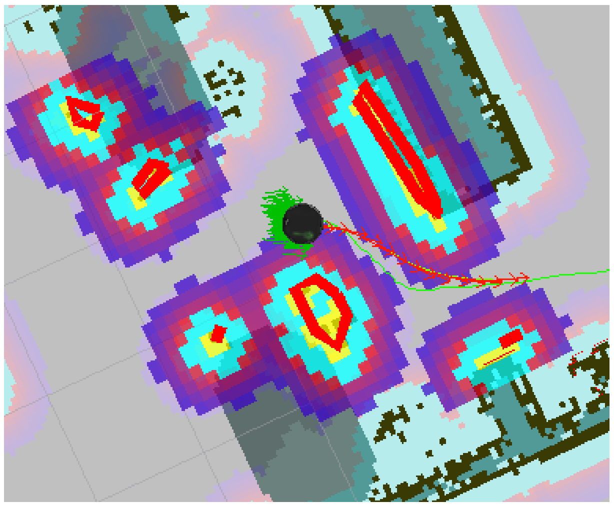


Fig. 4. Costmap converted by CostmapToPolygonsDBSMCCH plugin, vizualized with RVIZ tool. Yellow cells represents occupied cost cell, red polygons are converted clusters [26].

shapes are more suitable for describing obstacles such as a human or another robot, since all occupied points of such obstacles must be completely inscribed in the resulting polygon on the cost map in order to reduce the likelihood of collision. CostmapToPolygonsDBSConcaveHull plugin converts occupied cells to a set of concave polygons previously converting them into convex polygons using the DBSCAN algorithm [30]. Conversion of occupied cells into points and straight lines is possible using the CostmapToLinesDBSMCCH and CostmapToLinesDBSRANSAC plugins, with clusters generation also based on DBSCAN and RANSAC approaches.

V. Virtual Environment Setup

All experiments were conducted on a virtual building floor with a variety of rooms of different sizes and corridors of different widths using the Turtlebot3 Burger and Turtlebot3 Waffle robots. Before the experiments, a map of the virtual room was constructed using gmapping ROS package [31], [32]. Localization of the robot on the map is carried out using the AMSL, using this method, the most probable position of the robot is determined based on the set of its probable positions [25].

On the virtual map, in the rooms there are station signs that imitate the position of some objects with which the robot will interact in a hospital environment. In addition to stations, the simulation includes other moving people, robots, and a number of static obstacles such as furniture or rubbish. The robots are equipped with a 360°LiDAR sensor for localization and navigation, and the whole process of the robot's functioning relies only on this sensor.

VI. Virtual Experiments

At the start of virtual experiments, multiple Turtlebot3 Burger and Turlebot3 Waffle robots are located at random positions on the virtual map. The stations are located in the map rooms, mainly in the corners, and for visual purposes

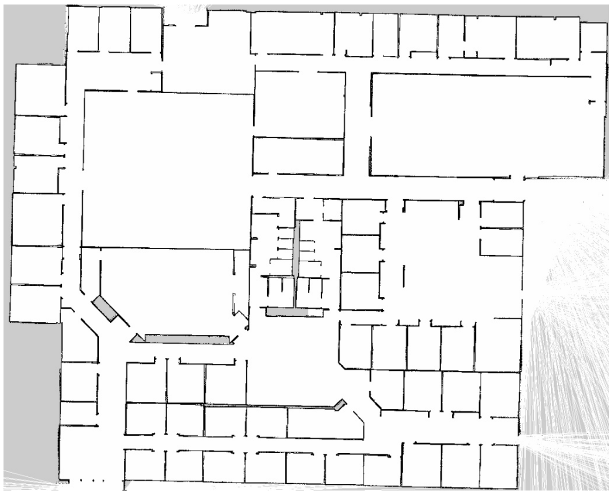


Fig. 5. The virtual environment model map constructed with gmapping algorithm.

they are presented as helipad models located outside the visibility zone of the robot's sensors (above the simulation surface). The initial state of the robot from the state machine is WAIT_FOR_GOAL. The input data for starting the execution of the robot is the Task data structure, which is described in more detail in Table 1. Tasks are assigned to the robot using the user interface with the choice of task parameters and choice of the robot responsible for the task execution.

After receiving the task, the robot enters the nested Task automaton with the NAVIGATE_TO_GOAL state and moves to the task assigned station.

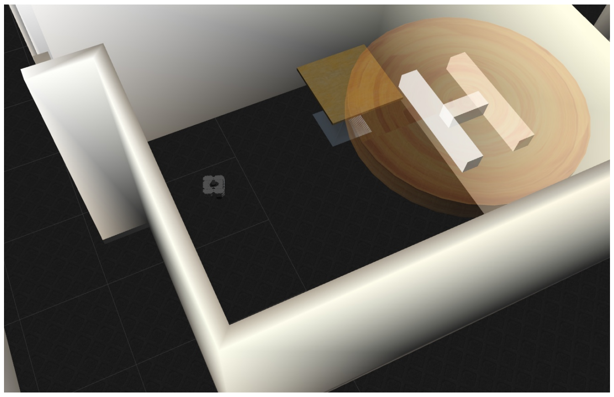


Fig. 6. TurtleBot3 Waffle is approaching the station in Gazebo simulation.

Upon reaching the station, the robot enters the EXECUTING_TASK state and, depending on the type of task, either waits for the time specified in the waitTime parameter, or waits for the standard value of the task execution time, after which its state exits to the original state machine and transitions to the WAIT_FOR_GOAL state, being here before receiving next task (if there are already tasks in the robot's task queue, then the robot will immediately go to the nested Task state machine with the initial state NAVIGATE_TO_GOAL).The robot stores information about the queue of its tasks, about stations and

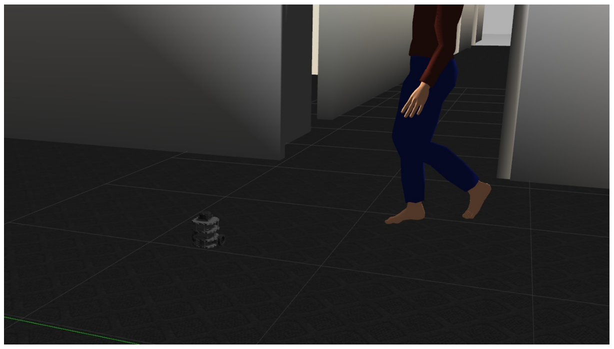


Fig. 7. TurtleBot3 Burger and moving human model in Gazebo simulation.

their positions on the map and the history of its completed tasks.

Acknowledgment

This work was supported by the Russian Foundation for Basic Research and the Cabinet of Ministers of the Republic of Tatarstan, project ID 18-48-160037.

References

[1] E. Magid, A. Zakiev, T. Tsoy, R. Lavrenov, and A. Rizvanov, "Automating pandemic mitigation," *Advanced Robotics*, pp. 1–18, 2021. [Online]. Available: https://doi.org/10.1080/01691864.2021.1905059

[2] G. Fragapane, C. Zhang, F. Sgarbossa, and J. O. Strandhagen, "An agent-based simulation approach to model hospital logistics," *Int J Simul Model*, vol. 18, no. 4, pp. 654–665, 2019.

[3] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida, "Developing a mobile robot for transport applications in the hospital domain," *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 889–899, 2010.

[4] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[5] ——, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots*. IEEE, 2013, pp. 138–143.

[6] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[7] K. Zakharov, A. Saveliev, and O. Sivchenko, "Energy-efficient path planning algorithm on three-dimensional large-scale terrain maps for mobile robots," in *International Conference on Interactive Collaborative Robotics*. Springer, 2020, pp. 319–330.

[8] W. K. Fung, Y. Y. Leung, M. K. Chow, Y. H. Liu, Y. Xu, W. Chan, T. W. Law, S. K. Tso, and C. Y. Wang, "Development of a hospital service robot for transporting task," in *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, vol. 1, 2003, pp. 628–633 vol.1.

[9] S. D. Klee, G. Gemignani, D. Nardi, and M. Veloso, "Multi-robot task acquisition through sparse coordination," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2823–2828.

[10] F. S. Melo and M. Veloso, "Decentralized mdps with sparse interactions," *Artificial Intelligence*, vol. 175, no. 11, pp. 1757–1789, 2011.

[11] Y. Sun, B. Coltin, and M. Veloso, "Interruptible autonomy: Towards dialog-based robot task management," in *Proc. Workshop 27th AAAI Conf. Artificial Intelligence*, 2013.

[12] J. Bohren and S. Cousins, "The smach high-level executive [ros news]," *IEEE Robotics Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[13] R. Mishra and A. Javed, "Ros based service robot platform," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, 2018, pp. 55–59.

[14] S. Pütz, J. Santos Simón, and J. Hertzberg, “Move base flex a highly flexible navigation framework for mobile robots,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3416–3421.

[15] R. Lavrenov, F. Matsuno, and E. Magid, “Modified spline-based navigation: guaranteed safety for obstacle avoidance,” in *International Conference on Interactive Collaborative Robotics*. Springer, 2017, pp. 123–133.

[16] E. Magid, R. Lavrenov, and I. Afanasyev, “Voronoi-based trajectory optimization for ugv path planning,” in *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*. IEEE, 2017, pp. 383–387.

[17] R. Yagfarov, M. Ivanou, and I. Afanasyev, “Map comparison of lidar-based 2d slam algorithms using precise ground truth,” in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1979–1983.

[18] E. Mingachev, R. Lavrenov, T. Tsoy, F. Matsuno, M. Svinin, J. Suthakorn, and E. Magid, “Comparison of ros-based monocular visual slam methods: Dso, ldso, orb-slam2 and dynaslam,” in *International Conference on Interactive Collaborative Robotics*. Springer, 2020, pp. 222–233.

[19] N. Alishev, R. Lavrenov, K.-H. Hsia, K.-L. Su, and E. Magid, “Network failure detection and autonomous return algorithms for a crawler mobile robot navigation,” in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2018, pp. 169–174.

[20] C. Rösmann. Teb_local_planner ros package web page. [Online]. Available: http://wiki.ros.org/teb_local_planner

[21] C. Rösmann, F. Hoffmann, and T. Bertram, “Integrated online trajectory planning and optimization in distinctive topologies,” *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[22] ——, “Planning of multiple robot trajectories in distinctive topologies,” in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.

[23] ——, “Kinodynamic trajectory optimization and control for car-like robots,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5681–5686.

[24] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, “Robust monte carlo localization for mobile robots,” *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[25] B. P. Gerkey. Amcl ros package web page. [Online]. Available: http://wiki.ros.org/amcl

[26] C. Rösmann. Costmap_converter ros package web page. [Online]. Available: http://wiki.ros.org/costmap_converter

[27] D. V. Lu, D. Hershberger, and W. D. Smart, “Layered costmaps for context-sensitive navigation,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 709–715.

[28] E. Marder-Eppstein. Costmap_2d ros package web page. [Online]. Available: http://wiki.ros.org/costmap_2d

[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[30] J.-S. Park and S.-J. Oh, “A new concave hull algorithm and concaveness measure for n-dimensional datasets,” *Journal of Information science and engineering*, vol. 28, no. 3, pp. 587–600, 2012.

[31] B. Gerkey. Gmapping ros package web page. [Online]. Available: http://wiki.ros.org/gmapping

[32] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid, “Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7226–7232.