Prioritizing Tasks Within a Robotic Transportation System for a Smart Hospital Environment

Ruslan Safin[1], Roman Lavrenov[1](✉), Tatyana Tsoy[1], Evgeni Magid[1], Mikhail Svinin[2], Sumantra Dutta Roy[3], and Subir Kumar Saha[4]

[1] Laboratory of Intelligent Robotics Systems (LIRS), Intelligent Robotics Department, Institute of Information Technology and Intelligent Systems, Kazan Federal University, Kazan, Russia
{lavrenov,tt,magid}@it.kfu.ru

[2] Information Science and Engineering Department, College of Information Science and Engineering, Ritsumeikan University, 1-1-1 Noji-higashi, Kusatsu, Shiga 525-8577, Japan
svinin@fc.ritsumei.ac.jp

[3] Department of Electrical Engineering, Indian Institute of Technology Delhi, IIT Campus, Hauz Khas, New Delhi, Delhi 110016, India

[4] Department of Mechanical Engineering, Indian Institute of Technology Delhi, IIT Campus, Hauz Khas, New Delhi, Delhi 110016, India
sumantra@ee.iitd.ac.in, saha@mech.iitd.ac.in

https://kpfu.ru/eng/itis/research/laboratory-of-intelligent-robotic-systems,
http://en.ritsumei.ac.jp/academics/college-of-information-science-and-engineering/,
http://ee.iitd.ac.in/, http://mech.iitd.ac.in/

Abstract. This paper describes a design and an implementation of a small-scale robotic transportation system, which operates in a smart hospital environment. Within a proposed framework unmanned ground vehicles (UGV) perform transportation tasks between multiple stations that are located in different rooms. The UGVs navigate in the environment with moving objects in accordance with basic traffic rules, which consider priorities of particular tasks of each UGV. UGVs' behavior is defined by a state machine and transitions between these states, which allows to make the robots' behavior more predictable and controllable. Virtual experiments were carried out in a simulation of an entire floor of a small-size hospital building using the Gazebo simulator. The experiments confirmed that using various task priorities shorten a path length of robots with high priorities and thus reduce their task execution time.

Keywords: Smart hospital · UGV · Robot navigation · Path planning · Task-based robotic system · Service robots · Leading edge healthcare

A. Ronzhin et al. (Eds.): ICR 2021, LNAI 12998, pp. 182–193, 2021.
https://doi.org/10.1007/978-3-030-87725-5_16

1 Introduction

Along with industrial robots, service robots need to maintain accuracy and speed while performing various operations. In contrast to industrial robots, service robots operate in the same space with humans and other robots. This requires robots to operate according to some predefined rules to ensure predictability of the behavior and thus a more productive joint work with a human staff [11,23].

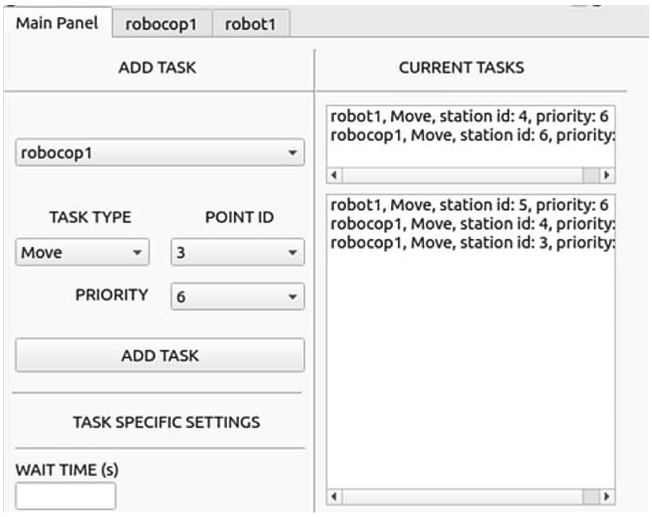
In the context of hospital environments, a transportation of objects is one of daily routines that could be delegated to service robots [16]. Currently, most transportation related tasks are performed manually and thus are human resource consuming, e.g., a patients' caring process involves delivering and collecting various objects several times a day. In turn, an automated transportation system (based on mobile service robots) allows to liberate hospital staff from transportation tasks [26] and benefits from a larger transport capacity, a possibility to optimize delivery routes, centrally assign priorities in task performance, and thus schedule a delivery order [8,9].

The paper is organized as follows. Section 2 summarizes related research including review of task based architecture, task manager and its GUI. In Sect. 3 a state machine structure and its implementation using the SMACH library is described [5]. Section 4 outlines robot movement rules, the corresponding states and transition conditions. Sections 5 and 6 describe a virtual environment setup for the system testing and experimental results. We conclude in the last Section.

2 Related Work

The main task of service robots, which are increasingly used in hospitals, is an automation of items delivery, which frees up staff time and increases an overall efficiency of such organizations [18]. One approach suggested using a mobile robot designed to automate delivery of medical items from one ward to another in a hospital [10]. The mobile platform Nomadic XR4000 used fluorescent lights of ceilings in order to determine its position and orientation. In [13] authors proposed a solution to improve the logistics of delivering items by robots by reducing a path traveled by the robots and task scheduling schemes, named deep Hungarian (d-Hungarian) and deep Voronoi (d-Voronoi). Paper [24] presented a task management problem for a mobile robot operating in an environment with humans and receiving new tasks from them. In our paper, we focus on an excessive autonomy of service robots, which implies a typical lack of a mechanism to interrupt (and control) ongoing tasks that are currently executed by the robots.

BačÍK et al. [3] described an approach, which included development of both hardware and software for a robot that performs delivery tasks in hospitals. It included a development of a powerlink interface to transfer data between a robot and a powerlink-compatible hardware, and improved a local path using Pure Pursuit Path Tracking Algorithm, which made initial path smoother and created a more continuous movement of the robot [4,7].



robocop1

TASK TYPE: Move

POINT ID: 3

PRIORITY: 6

WAIT TIME (s):

robot1, Move, station id: 4, priority: 6
robocop1, Move, station id: 6, priority:

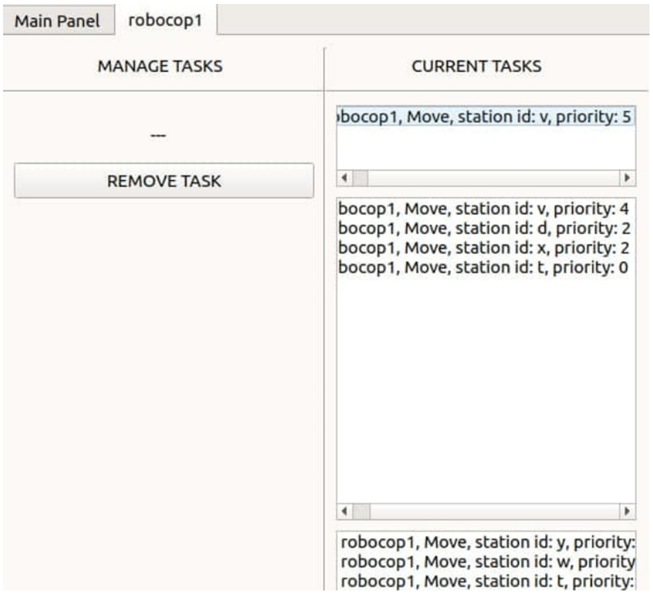
robot1, Move, station id: 5, priority: 6
robocop1, Move, station id: 4, priority:
robocop1, Move, station id: 3, priority:

Fig. 1. Main menu of the task manager GUI.

3 Tasks with Priorities Scheduler Implementation

We designed a system that assumes robots being engaged in transportation tasks in accordance with a list of tasks assigned to a robot and priorities of these tasks. Tasks are assigned via a graphical user interface (GUI). The GUI allows to assign a station (position) identifier to a task and to select an identifier of a robot that will perform the task. Each station corresponds to a particular position on a hospital map. Optionally, it is possible to set a waiting time after arriving to a station. When leaving “wait time” field blank, the robot’s state changes to WAIT_FOR_GOAL immediately upon a successful arrival to the task station. The task manager stores existing active tasks as queues (separately for each robot) and transfers started and completed tasks into appropriate states. Robots notify the manager when they start or finish a task.

The GUI main menu is shown in Fig. 1. A top panel is a list of tabs consists of a main menu tab and tabs for each robot. To display a status of tasks for an individual robot, tabs are provided for each launched robot. An example of a tab for a particular robot is shown in Fig. 2. The left side of the main menu contains functions for assigning tasks. The selection of all parameters (except the optional “wait time” parameter) is implemented as a drop-down list. A drop-down list of available robots is filled according to a list of names of robots to be launched. A station list is a set of station identifiers with station coordinates associated with each identifier. A priority list is a simple set of numbers from 0 to 9 that determines a position of a task in a robot’s task queue. A priority of a task executed by a robot also determines the priority of a robot in the movement rules that the robots follow when navigating in a hospital.



ɔbocop1, Move, station id: v, priority: 5

bocop1, Move, station id: v, priority: 4
bocop1, Move, station id: d, priority: 2
bocop1, Move, station id: x, priority: 2
bocop1, Move, station id: t, priority: 0

robocop1, Move, station id: y, priority:
robocop1, Move, station id: w, priority
robocop1, Move, station id: t, priority:

Fig. 2. Robot panel of the task manager GUI.

A task workflow works as follows. After a user assigns a task, the task is stored in the task manager's list of all tasks and is sent to a robot (to which the task is assigned). Each robot has its own task queue, which is updated each time a new task is assigned. When the robot receives a new task, the task is stored in the robot's queue and the queue tasks are reordered according to their priorities. The robot always executes the highest priority task from its queue. The task manager keeps track of all task states by receiving updates when the robot changes its task state to another. All assigned tasks are displayed in the GUI having either ongoing, pending (queued) or completed status. A task is considered completed when the robot has successfully arrived to its designated station and, depending on a presence of the wait_time parameter, has waited there for a specified time. After these conditions are met, the robot starts executing a next task from the queue, having previously informed the manager of a completion of a current task and a start of a new one. The tasks are stored as objects. A structure and a description of task fields are shown in Table 1.

4 Movement Rules Model

4.1 State Machine Implementation with SMACH

An overall behavior of a robot is defined by a finite state machine with various transitions from one state to another. The transitions depend on outcomes of a current state. Each state has its own set of outcomes, so that each state

Table 1. Task object fields description.

Field	Description
id	Unique identifier
isDone	Boolean value, set to true when a task is completed
isCurrent	Boolean value, set to true when a task execution starts
goalId	Unique identifier of a station on a map
taskType	Determines a robot task
robotName	Determines to which robot the task is assigned
waitTime	Optional value, a time that robot should wait after a task is completed

termination is accompanied by a transition to another state in accordance with an outcome of a previous state. The finite state machine structure was implemented by using the SMACH package, which is freely available and compatible with the Robot Operating System (ROS) framework [20]. The SMACH package provides a task-level architecture for a complex robot behavior that 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 at runtime, etc. This approach facilitates a task of controlling a robot behavior by decomposing its intended behavior into corresponding states, and allows the robot's states to be handled using data transitions between the states and conditions under which the robot changes its behavior. Table 2 lists the implemented robot states and their descriptions. The structure of the implemented state machine is shown in Fig. 3 as a visualization provided by the package smach_viewer [6]. WAIT_FOR_GOAL and NAVIGATE_TO_GOAL as active states are shown on the left and right sides, respectively.

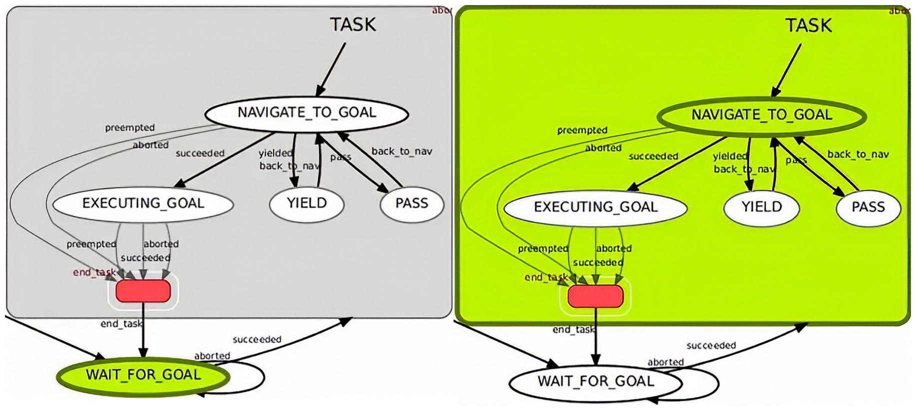


Fig. 3. SMACH active states smach_viewer package visualization.

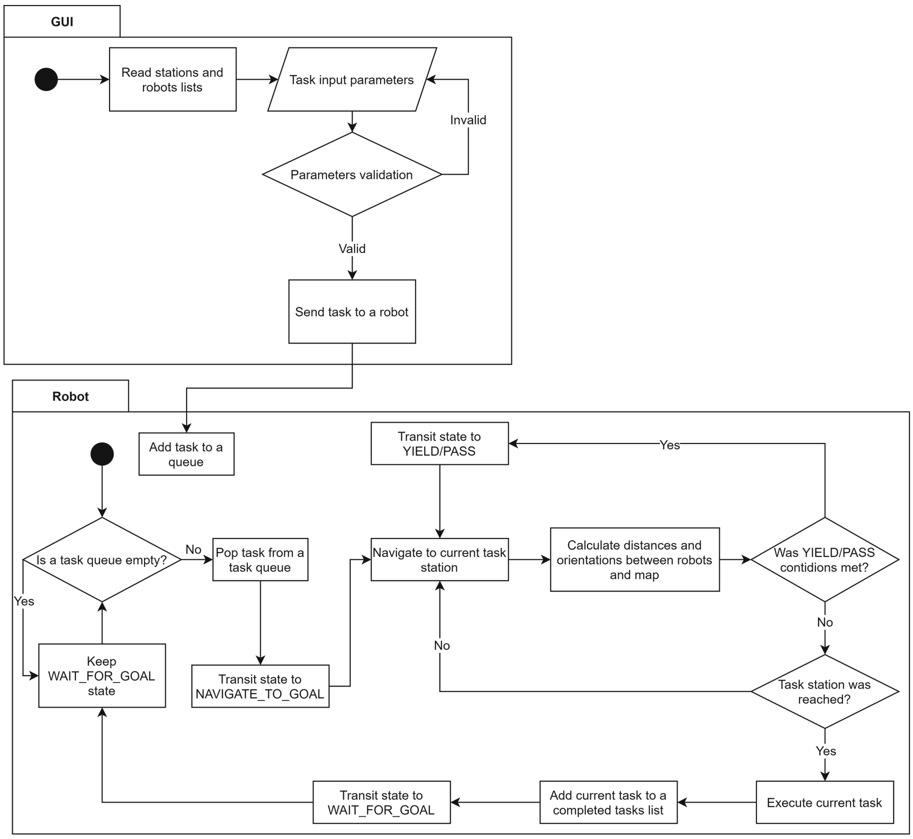

GUI
Read stations and robots lists
Task input parameters
Invalid
Parameters validation
Valid
Send task to a robot
Robot
Add task to a queue
Transit state to YIELD/PASS
Yes
Is a task queue empty?
No
Pop task from a task queue
Navigate to current task station
Calculate distances and orientations between robots and map
Was YIELD/PASS contidions met?
Yes
No
Keep WAIT_FOR_GOAL state
Transit state to NAVIGATE_TO_GOAL
No
Task station was reached?
Yes
Transit state to WAIT_FOR_GOAL
Add current task to a completed tasks list
Execute current task

Fig. 4. System block diagram.

Figure 4 shows a block diagram of the developed application. On launch, the task manager module reads the lists of stations and robots. They can be selected through the GUI. When the task options are selected and a task is sent, it is being passed and saved by a robot that is responsible for completing the task. The robot, initially in WAIT_FOR_GOAL state, transits to NAVIGATE_TO_GOAL state as soon as the new task appears in the list of its tasks. While navigating, conditions for transition to PASS and YIELD states are checked. In the case of transition to one of these states, a navigation to the task station is suspended until the robot returns to NAVIGATE_TO_GOAL state. As soon as robot reaches the station, its state changes to EXECUTING_GOAL state, which simulates a process of completing the task. A transition from this state implies a completion of the task, and the robot's state changes to the initial WAIT_FOR_GOAL state.

Table 2. Robot behaviour states description.

State	Description
NAVIGATE_TO_GOAL	Robot autonomously navigates to a point specified in a task
EXECUTING_GOAL	Robot executes goal, i.e. waits for a time specified by a user
WAIT_FOR_GOAL	Robot waits until a goal is received
YIELD	Robot gives a way to a higher priority robot (HPR): drive off to a side from HPR's direction
PASS	Robot gives a way to a higher priority robot (HPR): stop and wait until HPR passes

4.2 Describing Movement Rules with a Behavioral State Machine

While performing their tasks, robots should move from one station to another, ensuring the freest possible movement both for robots performing high-priority tasks and for humans (patients, guests, and staff), whose priority is always higher than that of the robots. To satisfy these conditions, a model of movement rules for robots was developed and implemented into the state machine. Currently, two additional states are implemented, each describing the robot's behavior when giving a way to a robot performing a task with higher priority. The list of implemented and tested rules for the movement is as follows:

1. When two robots approach each other, a robot with a lower priority task (lower priority robot, LPR), after approaching within 4 m, must give way to the second robot (higher priority robot, HPR), changing the direction of motion by 90 degrees relative to an HPR motion direction and moving 70 cm in that direction. The robots are considered to be moving toward each other when a difference between a rotation angle (yaw) of one robot relative to a map coordinate system and a rotation angle of the second robot relative to the same system is less than or equal to 0.27 rad and a position of one robot relative to the other along the X-axis (relative to the robot coordinate system) is positive, i.e., the robots are directed toward each other. After the HPR overtakes the yielding LPR (i.e., it is behind the point where the yielding LPR left its original route), the LPR continues moving towards its task. This rule corresponds to the state YIELD.
2. If two robots are moving in such a way that a LPR crosses the path of a HPR (e.g., a LPR exits a room into a corridor where another robot is traveling along, bypassing an exit from the room), the LPR must stop and wait for the HPR to move away. It is assumed that the robot routes intersect when vectors formed by two points (where the first point corresponds to coordinates of the robot's current position and the second is located at a distance of 2 m in the direction of the robot's movement) intersect and an absolute value of an intersection angle is in the range of 1.47 to 2.87. The LPR continues to move once the intersection condition is no longer satisfied. This rule corresponds to the state PASS.

The state YIELD is intended for the case when two robots move towards each other and one of them (a LPR) should give a way to a HPR. The robot state machine transitions to YIELD state when the conditions for the first movement rule are satisfied. As the robots continue to approach each other, their local planners begin to create a local path that considers another robot as an obstacle. Without appropriate movement rules this may fore the robots to move in parallel paths in the same direction (while trying to avoid each other) and deviate from their global paths. Since the conditions for entering this state imply that a robot is on the path of another robot, the behavior of the robot in the YIELD state is implemented to move sideways far enough in a way that the local planner of the HPR's path does not (or only insignificantly) change its local path plan, and thus does not significantly affect the execution time of the HPR's task.

The state PASS also characterizes avoidance behavior, but involves a full stop of a LPR under conditions where its continued motion will cross a local path of a HPR. The state machine transitions to YIELD state when the second movement rule conditions are satisfied. The LPR behavior in the PASS state is a full stop and wait until the HPR passes in front of the LPR or until the condition to transition to this state is no longer met.

In cases where multiple robots are in such situations at the same time, the same rules are followed, and LPRs begin to perform the actions of the YIELD or PASS states relative to the first HPR they encounter. When robots with equal priorities fall under these conditions, a robot with a smallest remaining path length to its station starts executing the YIELD/PASS behavior. If we assume that a robot with a larger distance to its station should yield, a scenario could be possible in which a robot located at a larger distance from its station would eventually be forced to significantly increase its estimated path since it gives a way to other robots with the same priority more frequently.

To work with the robot navigation, the navigation stack of the ROS framework was used, along with move_base, amcl [25], mapping [17], and other ROS navigation core packages [14,15]. TEB local planner was used as a path planner due to robustness and flexibility of it's tuning provided by it's parameters [21,22].

5 Virtual Environment Setup

Experiments were conducted in a Gazebo [1] simulated hospital provided by an aws_robomaker_hospital_world package [2]. The AWS hospital world depicts one floor of a building, which consists of a lobby with reception and a waiting area, staircases, various rooms including storage rooms, several patient wards and a staff break room. A top view of the hospital environment is shown in Fig. 5.

Station signs were placed in the rooms to represent locations of some objects that robots should interact with, or locations where they typically need to deliver items. Experiments were conducted with two TIAGo Base and one Clearpath Ridgeback robots. TIAGo Base is equipped with the Hokuyo URG-04LX-UG1 LIDAR sensor and Ridgeback is equipped with the Hokuyo UST-10LX LIDAR sensor, which is essential for performing localization and navigation.

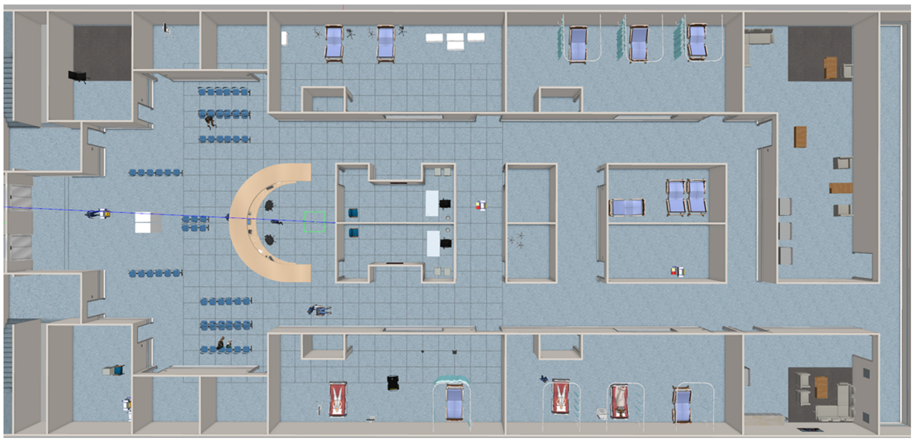


Fig. 5. Top down view of the Hospital World.

6 Virtual Experiments

At the beginning of the test runs, two TIAGo Base and one Clearpath Ridgeback robots were placed in different rooms of the virtual hospital; all state machines of the robots were in the state WAIT_FOR_GOAL. Next, the robots were assigned task sequences through the task manager GUI. The tasks had different priorities in a such way that the robots encounter each other while navigating to their stations and some robots would be forced to give way to others, i.e., at least once a transition from state NAVIGATE_TO_GOAL to states YIELD or PASS occurs. After the robots received the tasks, their state changed to NAVIGATE_TO_GOAL of the nested TASK state machine and they started navigating to the stations associated with tasks.

During the navigation, a LPR (with a lower priority task) encountered a HPR. An example visualized with the rviz [12] package is shown in Fig. 6. The paths built by the path planners are shown with curves: the yellow curve belongs to the Ridgeback and the red ones correspond to the TIAGo Base paths. Depending on the encounter conditions, the LPR (the TIAGo Base on the right), either transitioned to the YIELD state and made a way for the HPR (the Ridgeback on the left), or it transitioned to the PASS state and stopped to wait for the HRP to move further along its path.

Figure 7 shows two TIAGo Base robots freeing a way for the Ridgeback. The TIAGo Base approaching the higher priority Ridgeback entered the YIELD state, since the conditions for the first rule were met, and swerved aside to let it pass so that the local plan of the Ridgeback (yellow line) does not change. The local path of the TIAGo Base (red line above TIAGo Base on the right side of Fig. 7) changed accordingly to the rule YIELD. After the HPR traveled a sufficient distance, the yielding robot's state returned to NAVIGATE_TO_GOAL, and the robot continued towards its station. The experiment continued until

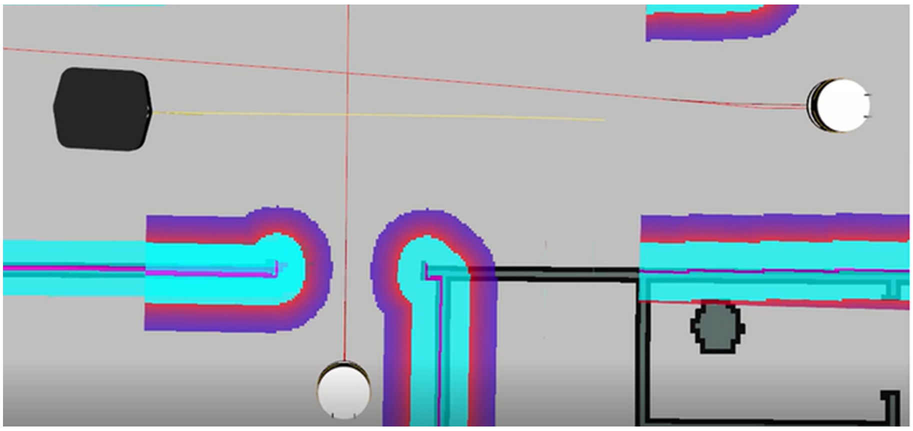


Fig. 6. The Clearpath Ridgeback (left) and TIAGo Base (right) robots are moving towards each other.

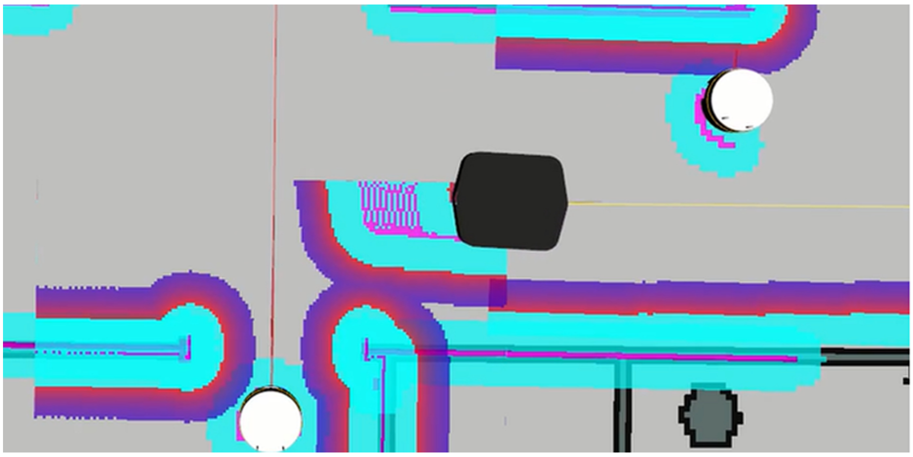


Fig. 7. The TIAGo Base robot in the active state YIELD gives way to the Ridgeback robot. (Color figure online)

all robots completed their tasks and returned to their initial state NAVIGATE_TO_GOAL.

7 Conclusion and Future Work

This paper presented the framework and the implementation of a small-scale robotic transportation system, which operates in a smart hospital environment. The unmanned ground vehicles (UGV) performed transportation tasks between multiple stations that were located in different rooms. The UGVs navigated in the environment with moving objects in accordance with basic traffic rules, which consider priorities of particular tasks of each UGV. UGVs' behavior was defined by a state machine and transitions between these states, which made the robots' behavior more predictable and controllable. Virtual experiments were carried out in the Gazebo simulation of an entire floor of a small-size hospital building. The experiments confirmed that using various task priorities shorten a path length of robots with high priorities and thus reduce their task execution time.

While the original system was designed for a dynamic hospital environment, it could be extended for other environments where the problem of labor shortage or performance arises. As a part of the future work we plan to expand UGV movement rules model with a human detection [19] and avoidance.

Acknowledgements. This work was supported by the Russian Foundation for Basic Research (RFBR), project ID 19-58-70002. The fifth author acknowledges the support of the Japan Science and Technology Agency, the JST Strategic International Collaborative Research Program, Project No. 18065977. This work is part of Kazan Federal University Strategic Academic Leadership Program. Special thanks to PAL Robotics for their kind professional support with TIAGo Base robot's Gazebo simulation related issues.

References

1. Abbyasov, B., Lavrenov, R., Zakiev, A., Yakovlev, K., Svinin, M., Magid, E.: 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), pp. 7226–7232 (2020). https://doi.org/10.1109/ICRA40945.2020.9196621
2. AWS robomaker: Amazon cloud robotics platform. https://github.com/aws-robotics/aws-robomaker-hospital-world
3. Bačík, J., Durovskỳ, F., Biroš, M., Kyslan, K., Perdukova, D., Padmanaban, S.: Pathfinder-development of automated guided vehicle for hospital logistics. IEEE Access **5**, 26892–26900 (2017)
4. Berntorp, K.: Path planning and integrated collision avoidance for autonomous vehicles. In: 2017 American Control Conference (ACC), pp. 4023–4028. IEEE (2017)
5. Bohren, J., Cousins, S.: The SMACH high-level executive [ROS news]. IEEE Robot. Autom. Mag. **17**(4), 18–20 (2010). https://doi.org/10.1109/MRA.2010.938836
6. Bohren, J.: SMACH smach_viewer package wiki page. http://wiki.ros.org/smach_viewer
7. Coulter, R.C.: Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie Mellon University (1992)
8. Fragapane, G., Zhang, C., Sgarbossa, F., Strandhagen, J.O.: An agent-based simulation approach to model hospital logistics. Int. J. Simul. Model. **18**(4), 654–665 (2019)
9. Fragapane, G., Hvolby, H.-H., Sgarbossa, F., Strandhagen, J.O.: Autonomous mobile robots in hospital logistics. In: Lalic, B., Majstorovic, V., Marjanovic, U., von Cieminski, G., Romero, D. (eds.) APMS 2020. IAICT, vol. 591, pp. 672–679. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57993-7_76

10. Fung, W.K., et al.: Development of a hospital service robot for transporting task. In: Proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, vol. 1, pp. 628–633 (2003). https://doi.org/10.1109/RISSP.2003.1285647
11. Galin, R., Meshcheryakov, R.: Automation and robotics in the context of industry 4.0: the shift to collaborative robots. In: IOP Conference Series: Materials Science and Engineering, vol. 537, p. 032073. IOP Publishing (2019)
12. Kam, H.R., Lee, S.H., Park, T., Kim, C.H.: RViz: a toolkit for real domain data visualization. Telecommun. Syst. **60**(2), 337–345 (2015)
13. Kumar, B., Sharma, L., Wu, S.L.: Job allocation schemes for mobile service robots in hospitals. In: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1323–1326. IEEE (2018)
14. Lavrenov, R., Magid, E., Matsuno, F., Svinin, M., Suthakorn, J.: Development and implementation of spline-based path planning algorithm in ROS/Gazebo environment. Trudy SPIIRAN **18**(1), 57–84 (2019)
15. Magid, E., Lavrenov, R., Svinin, M., Khasianov, A.: Combining Voronoi graph and spline-based approaches for a mobile robot path planning. In: Gusikhin, O., Madani, K. (eds.) ICINCO 2017. LNEE, vol. 495, pp. 475–496. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-11292-9_24
16. Magid, E., Zakiev, A., Tsoy, T., Lavrenov, R., Rizvanov, A.: Automating pandemic mitigation. Adv. Robot. 1–18 (2021). https://doi.org/10.1080/01691864.2021.1905059
17. Open Robotics Foundation. Gmapping ROS package web page. http://wiki.ros.org/gmapping
18. Ozkil, A.G., Fan, Z., Dawids, S., Aanes, H., Kristensen, J.K., Christensen, K.H.: Service robots for hospitals: a case study of transportation tasks in a hospital. In: International Conference on Automation and Logistics, pp. 289–294. IEEE (2009)
19. Ramil, S., Lavrenov, R., Tsoy, T., Svinin, M., Magid, E.: Real-time video server implementation for a mobile robot. In: 2018 11th International Conference on Developments in eSystems Engineering (DeSE), pp. 180–185. IEEE (2018)
20. ROS Operating System. http://wiki.ros.org
21. Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F., Bertram, T.: Efficient trajectory optimization using a sparse model. In: 2013 European Conference on Mobile Robots, pp. 138–143. IEEE (2013)
22. Rösmann, C.: Teb_local_planner ROS package web page. http://wiki.ros.org/teb_local_planner
23. Sagitov, A., Gavrilova, L., Tsoy, T., Li, H.: Design of simple one-arm surgical robot for minimally invasive surgery. In: 2019 12th International Conference on Developments in eSystems Engineering (DeSE), pp. 500–503. IEEE (2019)
24. Sun, Y., Coltin, B., Veloso, M.: Interruptible autonomy: towards dialog-based robot task management. In: 27th AAAI Conference on Artificial Intelligence (2013)
25. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust Monte Carlo localization for mobile robots. Artif. Intell. **128**(1–2), 99–141 (2001)
26. Zakharov, K., Saveliev, A., Sivchenko, O.: Energy-efficient path planning algorithm on three-dimensional large-scale terrain maps for mobile robots. In: Ronzhin, A., Rigoll, G., Meshcheryakov, R. (eds.) ICR 2020. LNCS (LNAI), vol. 12336, pp. 319–330. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60337-3_31