Development of an Autonomous Mobile Towing Vehicle for Logistic Tasks

Cláudia Rocha[1], Ivo Sousa[1], Francisco Ferreira[1], Héber Sobreira[1], José Lima[1,2](✉), Germano Veiga[1,3], and A. Paulo Moreira[1,3]

[1] INESC TEC - INESC Technology and Science, Porto, Portugal
{claudia.d.rocha,ivo.e.sousa,francisco.a.rodrigues,heber.m.sobreira, germano.veiga}@inesctec.pt

[2] CeDRI - Research Centre in Digitalization and Intelligent Robotics, Polytechnic Institute of Bragança, Bragança, Portugal
jllima@ipb.pt

[3] Faculty of Engineering of University of Porto, Porto, Portugal
amoreira@fe.up.pt

Abstract. Frequently carrying high loads and performing repetitive tasks compromises the ergonomics of individuals, a recurrent scenario in hospital environments. In this paper, we design a logistic planner of a fleet of autonomous mobile robots for the automation of transporting trolleys around the hospital, which is independent of the space configuration, and robust to loss of network and deadlocks. Our robotic solution has an innovative gripping system capable of grasping and pulling non-modified standard trolleys just by coupling a plate. Robots are able to navigate autonomously, to avoid obstacles assuring the safety of operators, to identify and dock a trolley, to access charging stations and elevators, and to communicate with the latter. An interface was built allowing users to command the robots through a web server. It is shown how the proposed methodology behaves in experiments conducted at the Faculty of Engineering of the University of Porto and Braga's Hospital.

Keywords: Mobile robot · Autonomous driving · Trolley docking · Ergonomics

1 Introduction

Even nowadays, carrying high loads and performing repetitive tasks is majorly done manually by employees. In the particular case of hospital environments, humans are responsible for the transportation of meal trolleys, laundry carts and other logistic tasks. These procedures tend to be recurrent and performed during long distances. Adding this to the high weight of the cars, which could sum up to almost three hundred *kg*, makes the health integrity of the employees become at risk.

The routine of carrying heavy objects, performing repetitive forceful tasks and inadequate postures tends to cause musculoskeletal disorders. These are

M. F. Silva et al. (Eds.): ROBOT 2019, AISC 1092, pp. 669–681, 2020.
https://doi.org/10.1007/978-3-030-35990-4_54

injuries or disorders of the muscles, nerves, tendons, joints, cartilage, and spinal discs, e.g. sprains, back pain, carpal tunnel syndrome, hernias that could get worse or even persist longer due to the work environment or performing continuously these tasks. Therefore, the goal of ergonomics is to diminish stress and eradicate disorders associated with the excessive use of muscles, bad postures and recurring tasks [1,5]. Thus, advancing solutions based on autonomous robotic systems capable of performing these tasks, is an important step towards improving the overall health of human resources and also achieving more efficient distributions of the personnel. However, these systems must adapt well to the environments where they shall make the difference, especially concerning safety issues. In crowded and chaotic locations such as hospitals, this is not trivial to address.

In this paper, we describe the development of an autonomous robotic system which may be used in different indoor environments without modifying them. Moreover, the system is able to use standard trailers without significantly changing them, as it only requires them to have a simple low-cost plate attached. It has a fleet administrator that supervises, coordinates and creates a list of missions ordered by an employee through a graphical interface. The specific use case of meal transporting hospital trolleys in a medical environment is here demonstrated but the concept can be applied to several areas. Experiments were conducted in both controlled and real environments, in a collaboration with TRIVALOR/GERTAL/ITAU in Braga's Hospital. We believe that such system will allow to reduce the injuries previously mentioned, to automate processes and to conduct operators to specific tasks which cannot be performed by a robot.

The structure of this paper is the following: Sect. 2 presents the work regarding the application of mobile robots in hospital environments, more precisely in the context of transporting trolleys. The architecture overview of the presented robotic system is described in Sect. 3. Then, the modules referred in the System Architecture will have a more detailed explanation in each corresponding Subsection. Section 4 describes the experimental tests conducted to validate the robotic system and presents the results achieved. Finally, Sect. 5 concludes the paper and point out some directions regarding the future work.

2 Related Literature

The development of smart logistics systems means the development of independent and flexible external and internal logistics solutions. The forthcoming Industry 4.0 means an accelerating growth of the efficiency of production systems [15]. There are a huge number of application of AGVs (Automated guided vehicles) in industries. They are revolutionising the way manufacturers move goods between places, increasing efficiency. Hospitals can also be benefited when moving trolleys with meals and linens.

Robotics can help hospitals to maintain care workflows and give staff more time for patient care. There are several robotic approaches applied to hospital logistics, but most of them are used for edutainment activities [10] or to transport medication in specific conveyors. This research has been carried out for some years, being an example the HelpMate robot, that carries late meal trays, sterile supplies, medications, medical records, reports, samples, specimens and mail while exhibiting humanlike behavior as it navigates [7]. According to our knowledge, there are no available approaches that transport, grasp and pull non-modified heavy standard trolleys. Most approaches transport the goods inside the robot and do not attach to an external trolley, as they are expecting a very particular kind of car model. Some commercial examples are the Moxi robot that is an hospital robot assistant that helps clinical staff with non-patient-facing tasks like gathering supplies and brings them to patient rooms and delivers lab samples. The proposed low cost solution allows to tow a trolley without modifying it, by simply coupling a plate into the latter.

Well known industrial mobile robot manufacturers with the capability of transportation and manipulation, such as MIR [11] and Swisslog [20], offer a lot of products and add-ons for AGVs. Inspired on the solutions referred previously and in the state of the art from automotive and industrial environment (such as BMW [6] and kuka) and hospital mobile robotics such as SAVANT "Target-free" AGV Hospital Cart Transportation System [14] and EvoCart [2], it is desired in this paper to present a new developed solution oriented to hospital facilities to transport food and linens that uses a low cost gripping system that handles the non-modified heavy standard trolley and transports it.

3 System Architecture

The modular architecture presented in Fig. 1 shows the underlying combination of software and hardware modules, and also the way how the users are able to command the robots. The developed system is composed by the integration of independent subsystems, allowing to add or remove any component without affecting the rest of the system, using ROS [9]. Given the importance of keeping hospital management tasks beyond reach, the architecture was designed to allow one or more users to communicate with the robots via the hospital private network.

Users are going to interact with the robotic system through an user-friendly and intuitive graphical interface designed to adapt their work needs. This may be done using different hardware, such as laptops and mobile phones. To streamline this process, a web server was created as the user's central command distribution unit. Regarding the software modules included in the robots, Fig. 1 shows a subset of them, the most relevant ones, in order to facilitate comprehension and analysis of the system. Our goal is to develop a fleet of trolley-transporting robots for an hospital. Each robot is independent and has its own state. In the next subsections is described more precisely.

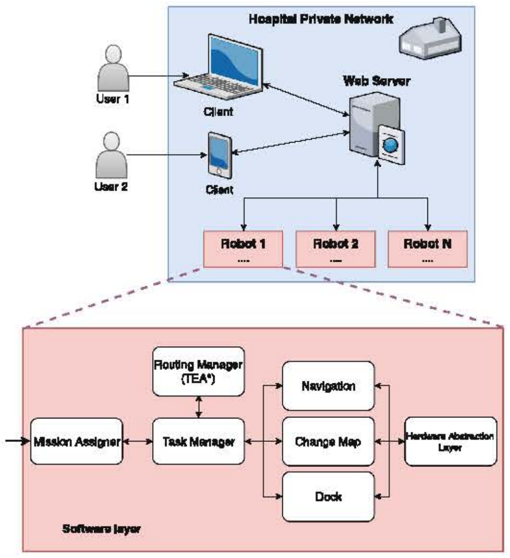


Fig. 1. System architecture diagram.

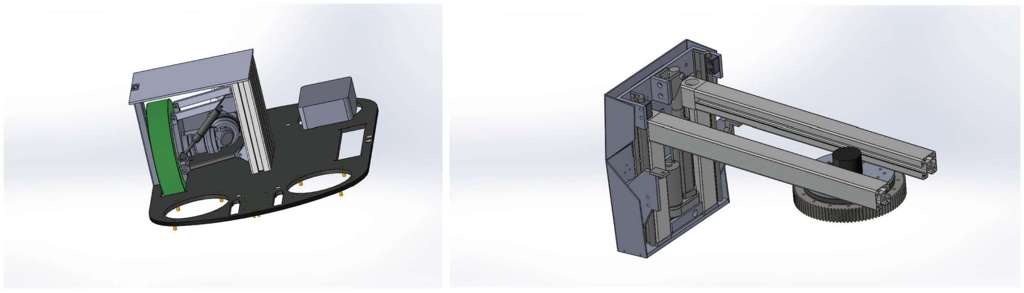
3.1 Hardware System

The robotic system comprises a mobile platform and a gripping system. The robot has a differential traction system with two spring-mounted drive-wheels with one motor each and four support wheels (Fig. 2a). It also has two Laser Range Finder SICK S300 which confer a 360° of view point and a bumper which incorporates a switch that is triggered on impact, allowing to detect obstacles and change behavior without sustaining damage. We assembled a tow arm on top of the robotic platform to act as a "hand" for grasping and pulling the trolleys (Fig. 2b). It consists of a structure which is just controlled by a linear actuator allowing two distinct movements. The orientation of the tow arm is known by a rotary absolute encoder of 10 bits. Arduino Mega was the microcontroller used to manage the trolley docking system. We considered a ROS node for monitoring connection and disconnection events, through Linux kernel event logs, allowing the system to be more robust to fails. Moreover, lighting and buzzing mechanisms were integrated to give feedback to the operator about the robot's state. To deal with a wide range of hospital car models, we have designed a simple, low-cost and easily integrable prototype plate for the docking task. This plate allows the robot to be attached rigidly to a trolley to be transported to its destination.

3.2 Human Robot Interaction

This module is responsible for enabling communications between users and the fleet of robots, in the sense that it allows users to define the operational behavior of the robots.

To control the robots, the user sends missions to them through a web application. As shown in Fig. 3, the user is allowed to send the robot from a given location to another one, to cancel one or all of the assigned displacements, and



(a) (b)

Fig. 2. Hardware components: (a) traction system; (b) gripping system.

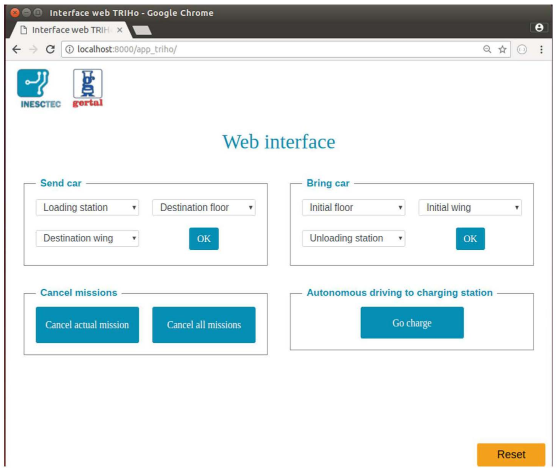


Fig. 3. Web interface prototype showing the fields where the user defines and cancels the missions.

to reset the state of the robot (by means of a joystick, the robot is sent to a predefined location, and reset is manually performed by pushing a button).

If any problem affects the server, disabling its proper operation, the robots may also be locally controlled without recurring to the external server, a state that is initiated after pushing specific physical buttons located on top of the robots. We are developing the web server using Django, a set of tools in python for streamline web development[1]. The web server provides the web page made in HTML and CSS in the browser of the device and the user responds to the forms. Then, the server receives the intended mission and sends it to the mission assigner. In reverse, the mission assigner gives feedback to the web server about the status of the mission. As currently implemented in our prototype, both communications are bidirectional using the HTTP protocol, differing in the type of data exchanged. In the deployment phase, we are switching HTTP for HTTPS, which is an extension of the first, where encryption allows the communications to be protected.

[1] https://www.djangoproject.com/.

3.3 Mission Assigner

The Mission Assigner module is the highest level software layer in the robot. This algorithm is responsible for receiving orders given by the user through the web interface and, autonomously, building the list of corresponding missions. For that, it is aware, in real time, of the current pose of the robot, the state of the battery and the arm state (actuated or not). Then, it decides, for example, if the robot needs to go charge itself or if new missions should be assigned. It also allows to cancel a single or all the missions stored in the robot. It is able to prioritize certain types of missions and to reset if any unexpected failure happens. This module communicates with the Task Manager through ROS Bridge Suite by using services and topics.

3.4 Action Planning: Time Enhanced A* (TEA*) and Token Manager

In a transportation system composed of a fleet of robots, it is important to choose the most optimized route for each one in order to prevent deadlocks between them. Nevertheless, unexpected events, such as the appearance of obstacles or delays during the execution of transporting tasks, need to be taken into consideration. Having this in mind, the Time Enhanced A* was chosen for the management of a fleet of robots. The TEA* Algorithm is used to determine a trajectory that a robot will have to perform for the coordination of multiple robots. It calculates each trajectory constantly, therefore it can be recognized as an on-line method. It is a path planning algorithm which resorts to a fixed graph to determine a trajectory based on the information of the 2D graph and the symbolic pose of each AGV (the edge each robot is occupying at the moment). A third dimension, *Time*, was added to enhance the information of the graph [12,13]. In other words, a temporal layer is created where each vertex can be considered as occupied or free. The robots perform each transportation task along edges, where the final point in each trajectory corresponds to a vertex in the graph. Therefore, this method searches by vertexes, where the cost function is associated to the length of the edges. As in the A* algorithm, that function results of the sum of the covered distance and the distance until the final point. The former represents the length of the edges and the latter the euclidean distance to the destination point. Figure 4 shows an example on how the TEA* works. From a fixed graph, where the red lines represent the edges, the blue circles represent the vertexes and the green arrow represents the desired orientation on each vertex, the algorithm determines the best route in order to reach a final point. In fact, that is represented by the yellow line over the chosen edges.

The token manager's module is a high-level supervision software, running in a central station, where it defines blocking and non-blocking areas in the trajectory to avoid deadlocks. It supervises, through UDP connections, the blocking areas' state (occupied or free), the last robot occupying that area and the real-time pose of each robot in the graph. Based on this information, the system is, additionally, able to deal with network fails.

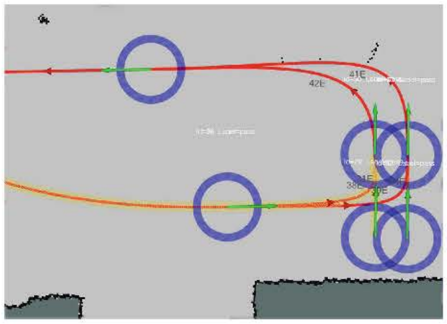


Fig. 4. Chosen trajectory by the TEA* (yellow) over a fixed graph. The red line represents the edges between vertexes (blue circle). The green arrow states the pretended orientation in each vertex.

3.5 Task Manager

At the robot level, Task Manager acts as an orchestrator, receiving top level missions from Mission Assigner and then breaking them into a sequence of specific tasks. These tasks are executed as ROS actions[2], in which the Task Manager interacts as the action client. For each action, there is a corresponding lower level action server node responsible for ensuring its execution, e.g., drive, dock and change map. Although each action has its own input parameters in order to execute the intended objective in a specific way, they are all executed similarly by the Task Manager, with an unified interface. This improves the modularity of the system, as it is easy to add or remove new functionalities (action servers).

As an exception, due to the higher behavioral complexity, the drive action is treated differently by the Task Manager. Before each one, the Task Manager interacts with the TEA* module, so it can receive the full path between the robot's current pose and the destination vertex. Then, the path is sent to the drive action which synchronizes its execution with the specific trajectory controller.

3.6 Navigation

The navigation system is composed by several modules, e.g. trajectory editor, differential/tricycle trajectory controllers, localization, map changer, which cooperate with each other so the robot can move around the environment.

For the localization process, the robot uses an extended version [4,19] of the Perfect Match algorithm [8]. It is a localization algorithm that does not require to change the target operation environment, as it uses natural features to estimate the robot's pose, e.g., walls, doors, cornerstones. It uses outlier rejection to be able to operate even in dynamic environments, when the input sensor data may be different than the previously created map. As seen in [16,17], the Perfect Match is a computationally light algorithm which performs better (considering

[2] http://wiki.ros.org/actionlib.

a 2D space) in terms of convergence speed, robustness to initialization errors, and has similar precision, comparatively with other PCL based localization algorithms.

In this specific application, the robot movement will be based in predefined fixed trajectories, as it is intended to have a high degree of predictability and repeatability, when sharing an environment with humans. So, by using a trajectory drawing software (based in parametric curves), all the possible paths will be defined, in a form of a graph, where the robot will be moving. Graphs are constituted by vertexes (possible stopping poses) and edges (parametric curves that connect two vertexes), as seen in Fig. 5. Each edge has its own forward and backwards linear velocities associated, which is constant when the robot is moving along that specific path. On the other hand, each vertex may have some actions associated with it, e.g., exchange map, interact with elevator or order to dock a trolley. The specific trajectory performed by the robot for each objective is calculated and optimized by the routing TEA* algorithm, better described in Sect. 3.4.

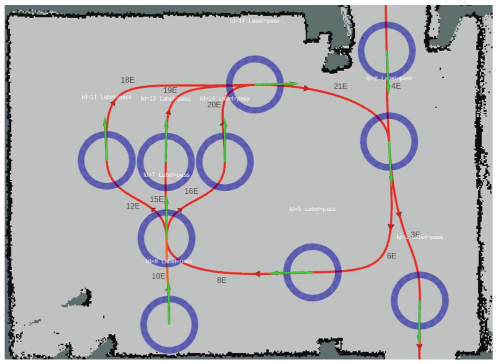


Fig. 5. Graph example. The red lines represent the edges and the blue circles represent the vertexes, with orientation given by the green arrows.

This robot base, as previously mentioned, has two traction wheels and the drive control is based on the differential traction model. When the robot is pulling the previously docked car, the robotic system exchanges to tricycle traction model, with the robot base acting as the direction wheel. This assures more precision and repeatability in the path following task and when some maneuvers are performed (more critical when driving backwards), e.g., parking the trailer and entering elevator.

This robot system ensures multi-floor navigation. As such, after mapping the robot's target environment (e.g., multi floor building), all maps are saved to be later used by the robot. There is a specific module responsible for coordinating the current navigation map with the robot's tasks, using all previously saved maps and the transformations between them.

3.7 Docking

In this section, a general overview on the different docking modes used on our application is presented. The robot performs a docking task to the elevator in order to change between floors, to reach the cars for their transportation and to move to the charger for the autonomous charging of the robot.

Elevator. The process responsible for the robot entering and exiting an elevator is controlled by a specific module, where the Perfect Match algorithm is used by the robot itself to find its location regarding the object of interest. A small map of the elevator contours is used, in order for the robot to estimate its pose in relation to the elevator. Then, when the clear space in the elevator is identified, the robot can generate an on-line trajectory in order to reach a predefined pose inside the elevator. However, it can also use an already existing path of the graph. This is done both when the robot is operating in differential and tricycle modes (with a trailer attached to its arm). It is possible for the robot to enter and exit the elevator either forward or backwards.

Car and Charger. We resorted to a Beacon-based Localization Algorithm [3,18], for the detection of the trolley or the charger. This algorithm is divided in several modules, for example the first one is responsible for the Kalman filter which handles the sensor fusion with the odometry. Other modules try to use the data from a laser range-finder to determine a possible position of cylindrical beacons and try to identify them. However, the usage of cylindrical beacons is unsuitable for our application due to the fact of their three dimensional shape, they can be easily damaged or moved and are voluminous, which led us to choose planar beacons. For this system to work, it needs as input: (a) the pose of the car we want the robot to dock, (b) the pose of the robot and (c) the detected beacons by the laser range finder. As the measured distance between the beacons attached to the car or the charger are known, we can group the detected ones in pairs. Each group may represent a possible pose of the object we want the robot to dock. This will allow to discard false beacons as well as give a more precise pose estimation of the car or charger to the Kalman Filter that processes the Beacons Localization Algorithm (Fig. 6a). With the poses as an input, we compute the theoretical pose of the car referenced to the robot. If this matches with a pose estimated by the detected beacons, the docking process is initialized. During the docking process, the robot will follow a line to approach the docking pose while testing if both beacons are visible through the whole trajectory (Fig. 6b).

When the robot reaches the final pose, a final verification is performed. In fact, it is determined if there is any deviation of the pose of the robot towards the pose of the car. If this verification is successful, a message is emitted to the upper level, which sends the signal for the arm to descend.

During the docking process, certain situations may lead to a failure: (a) the pose estimated by the detected beacons does not match the expected pose, (b) one or both beacons are not detected while the robot is approaching the trolley

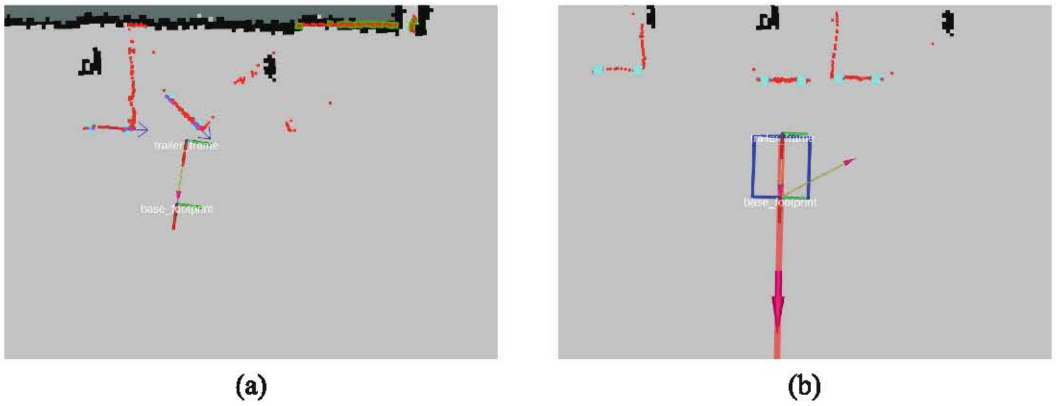


Fig. 6. Docking process: (a) detection of the beacons (cyan squares) and pairing (blue arrow); (b) robot following a line in the docking process.

or the charger and (c) the failure of the final verification. When any of these scenarios occurs, an error message is emitted to the upper level, starting a retry process.

4 Experiments and Results

We conducted a set of experiments to evaluate our system, which can be seen in the following link: https://youtu.be/TMnIOADs-dM. We tested each module separately, e.g. we instructed the robot to dock a trolley (blue and orange block) or an elevator, to drive to a specific destination with the car docked (violet and red) and also without it. Upon the conclusion of these preliminary tests, the system was tested as a whole. In this last experiment (Fig. 7), the robot docked the trolley and carried it from a floor to another, reached its destination, undocked the car and returned to the starting point (green block). When the robot arrived, a mission to return the car to its original position was sent.

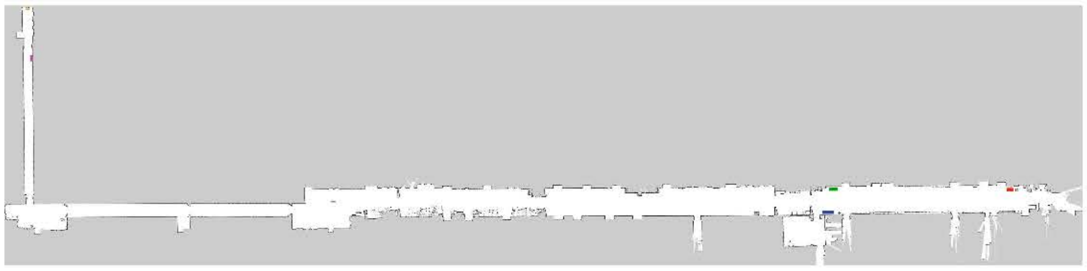


Fig. 7. Map of the ground floor of Braga's Hospital. The blue and orange blocks represent loading areas, the violet and the red blocks represent the unloading areas and the green block represent the charging station's area and the base position.

In order to evaluate the robotic system performance, several experiments were made keeping the same sequence of the operations. Previously, we set up a multi-

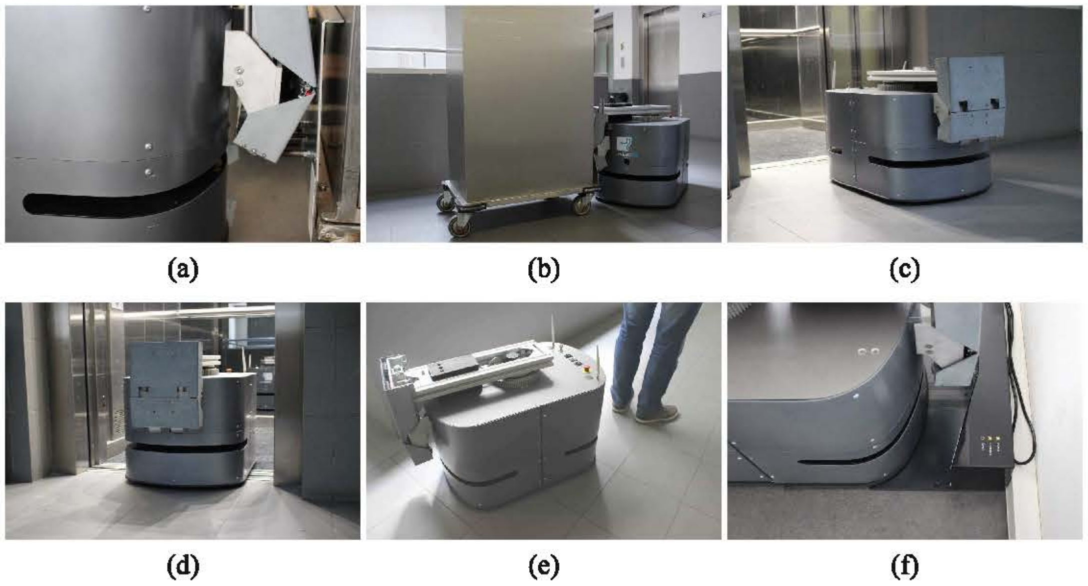


Fig. 8. Snapshots taken during the experiments: (a) robot docks the trolley; (b) robot navigating to a specific vertex; (c) robot docking the elevator; (d) robot undocking the elevator; (e) robot stops if an object or a person is detected in its security area; (f) robot charging.

floor trajectory involving communications with an elevator. Then, we provide a list of missions through a web interface. The robot fetched the trolley by docking it (Fig. 8a), went to the vertex before the elevator (Fig. 8b) and sent an order to the elevator to move to that floor. When it arrives, the robot starts the docking process to the elevator (Fig. 8c) and the elevator goes to the desired floor. After, the door opens and the robot leaves the elevator (Fig. 8d). It follows its path and, if any obstacle appears, the robot detects it and stops (Fig. 8e). The robot concludes its mission by un-docking the car at a predefined location. As soon as the mission finishes, it charges by docking to the charging station (Fig. 8f). All the experiments were conducted in the Faculty of Engineering of the University of Porto and in Braga's Hospital. To evaluate our system, we performed various tests in distinct routes and we used diverse obstacles to assure its flexibility. The outcome of these experiments was successful.

5 Conclusions and Future Work

This paper described the development of an autonomous robotic system for the transportation of hospital trolleys in medical environment. It presented the software architecture of the system, which is divided in several modules. A human machine interface was developed to simplify the interaction with the system. The graphical interface, where a user can outline a sequence of missions to be executed was presented. This system can be used in other types of environments, e.g. inside factories, without major changes to the environment itself. To deploy this system in other facilities, a map of the new environment and a fixed graph are required. We preferred fixed trajectories over free navigation in order to achieve a more predictable system. This is important in our application, since the robot may transport heavy loads in crowded environments. The whole system is being tested in a controlled environment in the Faculty of Engineering of the University of Porto, as well as in an hospital environment. As stated in Sect. 4, our system behaved as expected, performing every demanded mission without any hazardous or unexpected behavior. Regarding future work, the coordination of multiple mobile robots on different environments will be addressed.

Acknowledgements. This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme, and by National Funds through the Portuguese funding agency, FCT-Fundação para a Ciência e a Tecnologia, within project SAICTPAC/0034/2015- POCI-01- 0145-FEDER-016418.

Authors would like to acknowledge to Trivalor, Itau and Gertal for the support of the project RDH.

References

1. Center for Disease Control and Prevention: Work-related musculoskeletal disorders and ergonomics (2018). https://www.cdc.gov/workplacehealthpromotion/health-strategies/musculoskeletal-disorders/index.html
2. EvoCart: Evocart - advanced motion technology (2019). https://www.oppent-evo.com/en/home-en/
3. Ferreira, F., Sobreira, H., Veiga, G., Moreira, A.: Landmark detection for docking tasks, pp. 3–13 (2018)
4. Gouveia, M., Moreira, A., Costa, P., Reis, L., Ferreira, M.: Robustness and precision analysis in map-matching based mobile robot self-localization (2009)
5. Iowa State University Department Environment Health and Safety: Ergonomics (2016). http://publications.ehs.iastate.edu/ergo/
6. Kochan, A.: BMW uses even more robots for both flexibility and quality. J. Ind. Robots (2005). https://doi.org/10.1108/01439910510600173
7. Krishnamurthy, B., Evans, J.: HelpMate: a robotic courier for hospital use. In: IEEE International Conference on Systems, Man, and Cybernetics (1992)
8. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: an efficient and accurate approach for robot self-localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005: Robot Soccer World Cup IX, pp. 142–153. Springer, Heidelberg (2006)
9. Martinez, A., Fernández, E.: Learning ROS for Robotics Programming. Packt Publishing, Birmingham (2013)
10. Messia, J., Ventura, R., Lima, P., Sequeira, J., Alvito, P., Marques, C., Carriço, P.: A robotic platform for edutainment activities in a pediatric hospital. In: 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) (2014)
11. MiR: Mobile industrial robots (2019). https://www.mobile-industrial-robots.com/en

12. Santos, J., Costa, P., Rocha, L., Moreira, A., Veiga, G.: Time enhanced A*: towards to the development of a new approach for multi-robot coordination. In: IEEE International Conference on Industrial Technology (ICIT) (2015)
13. Santos, J., Costa, P., Rocha, L., Vivaldini, K., Moreira, A.P., Veiga, G.: Validation of a time based routing algorithm using a realistic automatic warehouse scenario. In: Reis, L.P., Moreira, A.P., Lima, P.U., Montano, L., Muñoz-Martinez, V. (eds.) Robot 2015: Second Iberian Robotics Conference, pp. 81–92. Springer, Cham (2016)
14. Savant Automation: Savant automation - AGV systems (2019). http://www.agvsystems.com/
15. Skapinyecz, R., Illés, B., Bányai, A.: Logistic aspects of industry 4.0. In: IOP Conference Series: Materials Science and Engineering, vol. 448, no. 1 (2018)
16. Sobreira, H., Rocha, L., Costa, C., Lima, J., Costa, P., Moreira, A.P.: 2D cloud template matching-a comparison between iterative closest point and perfect match. In: 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 53–59 (2016)
17. Sobreira, H., Costa, C.M., Sousa, I., Rocha, L., Lima, J., Farias, P.C.M.A., Costa, P., Moreira, A.P.: Map-matching algorithms for robot self-localization: a comparison between perfect match, iterative closest point and normal distributions transform. J. Intell. Robot. Syst. (2018). https://doi.org/10.1007/s10846-017-0765-5
18. Sobreira, H., Moreira, A., Costa, P., Lima, J.: Robust mobile robot localization based on a security laser: an industry case study. Ind. Robot: Int. J. **43**, 596–606 (2016)
19. Sobreira, H., Pinto, M., Moreira, A.P., Costa, P.G., Lima, J.: Robust robot localization based on the perfect match algorithm. In: Moreira, A.P., Matos, A., Veiga, G. (eds.) CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control, pp. 607–616. Springer, Cham (2015)
20. Swisslog: Swisslog - healthcare and logistics automation (2019). https://www.swisslog.com/

Author Index

M. F. Silva et al. (Eds.): ROBOT 2019, AISC 1092, pp. 683–685, 2020.
https://doi.org/10.1007/978-3-030-35990-4