2019 IEEE Intelligent Transportation Systems Conference (ITSC), 27-30th October 2019, Auckland, New Zealand
DOI: 10.1109/ITSC.2019.8917469

Autonomous Navigation for Mobility Scooters: a Complete Framework Based on Open-Source Software*

Marco Cecotti[1] and Nabil Aouf[2]

***Abstract*— In recent years, there has been a growing demand for small vehicles targeted at users with mobility restrictions and designed to operate on pedestrian areas. The users of these vehicles are generally required to be in control for the entire duration of their journey, but a lot more people could benefit from them if some of the driving tasks could be automated. In this scenario, we set out to develop an autonomous mobility scooter, with the aim to understand the commercial feasibility of a similar product.**
This paper reports on the progress of this project, proposing a framework for autonomous navigation on pedestrian areas, and focusing in particular on the construction of suitable costmaps. The proposed framework is based on open-source software, including a library created by the authors for the generation of costmaps.

I. INTRODUCTION

The market for mobility scooters has seen a rapid growth over the past decade and further expansion is expected in the future [1]. This is partly due to the process of ageing of the world population [2], partly to governmental incentives aimed at improving the lifestyle of elderly and disabled people, and partly to the growing popularity of electric vehicles and the resulting availability of affordable and reliable components.

Many people with mobility impairment have benefited from the usage of mobility scooters, becoming more independent and socially active than otherwise possible[1]. Unfortunately, the same benefits are out of reach of people who do not have the mental and physical capabilities to reliably control the vehicle, because mobility scooters are generally not designed to assist in the driving task.

In recent years, a few papers have studied solutions to assist the users of mobility scooters with driving. *Bingham et al.* [3] proposed a collision avoidance system using infrared and ultrasonic sensors, while *Eck et al.* [4] presented a solution to guide the user through narrow passages. In both cases, the user was expected to guide the vehicle towards the desired destination.

A considerable number of studies is focused on the development of fully-autonomous systems on pedestrian areas. In Japan, there is even a competition involving robots on pedestrian areas open to the public [5]–[7]. Some of these robots are mobility scooters modified for the event [7]. *Morales et al.* [5], *Date et al.* [6] and *Hirai et al.* [7] presented very detailed descriptions of their designs, all relying on map matching for localisation. In general, these studies rely on binary maps to distinguish between traversable and non-traversable surfaces, with some features, such as road edges, used for localisation.

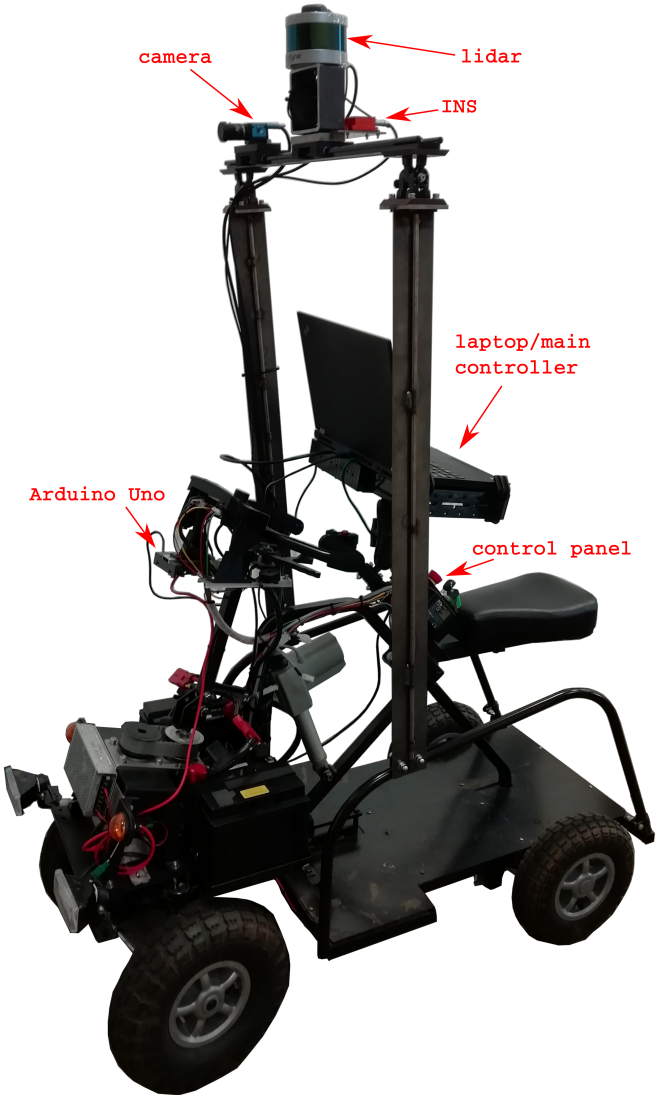

Fig. 1. The vehicle used for the demonstration

With the aim of improving the accessibility of mobility scooters, we set out to develop an autonomous prototype. The main goal of our project is to learn about the technical

*This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) and TMSUK Company Limited

[1]Marco Cecotti is with the Centre for Electronic Warfare Information and Cyber, Cranfield University, Defence Academy of the United Kingdom, Shrivenham, SN6 8LA, UK m.cecotti@cranfield.ac.uk

[2]Nabil Aouf is with the Department of Electrical and Electronic Engineering, City, University of London, Northampton Square, London, EC1V 0HB, UK nabil.aouf@city.ac.uk

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

challenges and to understand the commercial feasibility of a mobility scooter that requires minimal input from the user, and ideally no inputs at all during the journey. We aim to demonstrate outdoor navigation capabilities by delivering a vehicle that can:

- track its position with or without GNSS
- plan a path on pedestrian areas between two points in the map, favouring the usage of dedicated infrastructure (e.g. pavement instead of grass)
- follow the path smoothly, without hitting obstacles or falling off the kerb
- cross the road safely by checking that the road is clear
- move around obstacles if the path is partially blocked

The functionality of the vehicle was demonstrated around part of the university campus. The area is representative of a quiet urban area, and includes a good variety of features and surface types: road, pavement, lawn, intersections, pedestrian crossings, and kerbs.

The main contributions of this paper are:

1) A comprehensive description of the design of system and software architecture for an affordable mobility scooter with autonomous capabilities
2) The definition of a development framework completely based on open-source software, providing useful information on the limitations of the available libraries
3) A thorough presentation of the construction of costmaps suitable for autonomous navigation on pedestrian areas

This paper is organised as follows. In Section II, we describe the components used for autonomous navigation and their main specifications. In Section III, we present the software used in our project, providing particular detail on the construction of useful costmaps. Finally, in Section IV, we summarise the limitations of the proposed design and we define future research steps.

II. SYSTEM ARCHITECTURE

A. Base Vehicle and Actuators

The base vehicle is provided by TMSUK. It is a prototype version of the RODEM robot [8], modified for outdoor usage on public roads and pedestrian areas. The production version of the RODEM is designed to support remote operations via Bluetooth, so it features a well-proven set of components for traction and steering. Our prototype inherits the same system, powered by two lead acid batteries connected in series to supply a $24\,\mathrm{V}$ DC bus.

B. Controllers

The main controller is using a standard laptop, with a powerful processor (Intel Core i7 6820HQ) for fast calculations and a sizeable memory ($32\,\mathrm{GB}$) to process large data. The memory requirement is particularly important in the mapping process, when large and detailed point clouds are used.

An Arduino Uno with a bespoke I/O board is used to interface the traction motor controller and the steering motor controller with the laptop, through USB virtual COM.

C. Sensors

For localisation and perception of the environment we are using a Velodyne VLP16, a lidar sensor with a range of $100\,\mathrm{m}$ and 16 beams covering 360° around the vehicle. This sensor is mounted on a rail above the seat, as shown in Figure 1, in order to detect obstacles all around the vehicle with a single sensor. Unfortunately, due to the high location and a vertical field of view of only 30°, there is a large blind spot underneath the sensor, shaped as a cone with a base radius of roughly $7\,\mathrm{m}$.

Localisation and odometry are achieved with an XSENS MTi-G-710, an Inertial Navigation System (INS) consisting of a Global Navigation Satellite System (GNSS) receiver and a Inertial Measurement Unit (IMU) in a single package. The GNSS receiver is used to initialise the position estimate. After initialisation, the position can be tracked without the GNSS signal by matching the output of the lidar with a pre-recorded map.

The vehicle is also fitted with a high-resolution camera (The Imaging Source DFK 33UX264) to record the scene in front of the vehicle. At the moment the camera is not used for automated perception of the environment, but the camera output is recorded to support the analysis of the output of the other sensors. In the near future, however, we are planning to use images from the camera to extend the point cloud recorded by the lidar, increasing the point density and the area covered [9], [10].

III. SOFTWARE ARCHITECTURE

A. Operative System and Middleware

The choice of the operative system was critical for the evaluation of open-source libraries, in order to avoid porting of the code. Early in the project, we decided to focus on libraries compatible with the robotic middleware called Robot Operating System (ROS) [11], because of the popularity of this platform. Among several tools, a particular attention was given to Autoware [12], one of the few complete packages for autonomous navigation. In order to evaluate this tool, we decided to run our software on Ubuntu 16.04, the most recent version supported by Autoware at the time.

B. Sensor Interface

The Velodyne lidar and the XSENS INS are interfaced with the main controller using two ROS packages: *velodyne* [13] and *xsens_driver* [14]. Interfacing with the camera was slightly more challenging, due to the lack of direct support for ROS. Ultimately this problem was solved by using the drivers provided by The Imaging Source, a software called *gstreamer1.0* and the ROS package *gscam*, compiled to support gstreamer1.0.

C. Localisation and 3D Mapping

One of the goals of the project is to maintain an accurate localisation in case of loss of the GNSS signal, so in order to keep track of the vehicle position we have to rely on dead-reckoning and loop-closure [15] by matching the output of

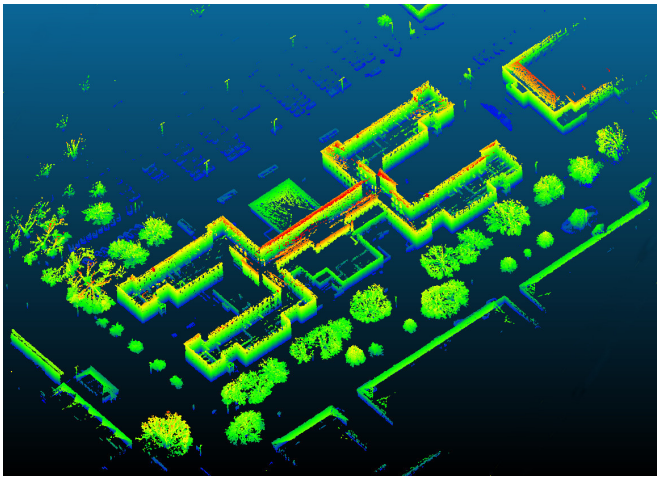

Fig. 2. The 3D map of the demonstration area

the lidar with a 3D map. In order to generate this map we need to match multiple scans of the lidar among themselves.

Autoware provides us with a robust solution for both of these matching problems, in the form of two software libraries: *ndt_mapping* is used to generate the map offline, while *ndt_matching* is a fast and efficient implementation to register the latest point cloud with the previously-calculated map. Both solutions register the point clouds in the form of Normal Distribution Transform (NDT) maps [16], [17]. The output of *ndt_mapping* is shown in Figure 2.

A thorough review of the performance of the mapping library is beyond the scope of this paper, but we can say that the results of the mapping process were qualitatively very positive. Despite starting from a difficult dataset, due to the usage of a low-resolution lidar and an unstable recording platform that experienced harsh roll oscillations, the matching algorithm performed very well, without the need to supply odometry information from external sources (e.g. the XSENS INS). By exploring the map with a visualisation tool, we were not able to detect any distortion. We also compared the result with satellite images available online (e.g. from Google Maps) and the accuracy of the map is on a par with the resolution of the images across the whole area ($215\,\mathrm{m} \times 118\,\mathrm{m}$).

The main issue with the library *ndt_matching* is the lack of raytracing to remove dynamic objects in the scene [18], if successive scans show that the object has moved. Because of this, we had to manually remove the trails left by dynamic objects using a point-cloud editing software called CloudCompare.

D. Costmap and Path Planning

Pedestrian areas are semi-structured environments [19] where the vehicle can travel in any direction across the traversable area, with preference over certain patches compared to others. This preference can be based on the user comfort of travelling on flat surfaces instead of uneven ground, on the desire to maintain the vehicle clean by avoiding mud, or to strictly respect rules by avoiding private properties. All of these preferences need to be scored against other targets, such as the desire to move or to reach the destination within a certain time, to allow the vehicle to use the non-preferential patches when this is beneficial for the user. An established way to score the movement across different areas is to use a costmap [20].

Costmaps are structures that assign a cost to each cell in the map. They are typically used to define the cost of a path, as the sum of the cost of the cells traversed by the path. In general, a useful costmap should account for dynamic objects, as they are perceived by the sensors, and a pre-recorded static environment, used to overcome the limitations of real-time perception (range, resolution, occlusions, object detection, classification, segmentation, etc.). A good way to represent this information is to use different costmap layers [20], organised according to the source of the data or the type of cost they represent. While these layers can update independently, they are ultimately aggregated into a master layer, which is the input to the path planning algorithm.

With the long-term goal of deploying autonomous vehicles on pedestrian area on a large scale, we briefly considered whether it is possible to generate costmaps automatically. Our opinion is that, while it might be possible to automatically detect some types of pavement [21], the varieties of surfaces that can be encountered on pedestrian areas is so vast that it is currently not feasible to automate the segmentation and the classification of most of them in a reliable way. In addition, the preference towards some patches is often subjective, so it would be hard to find a costmap that suits everyone in the same way. Because of the aforementioned reasons, we decided to look at a framework for a semi-automated generation of costmaps, that relies on manual segmentation of the area. This process is divided in two branches: the generation of a costmap from a manually-defined vector map, and the generation of a costmap from a 3D map. The Matlab code used to support the generation of costmaps will be made available under the MIT licence on Github[1].

1) From 3D map to 2D Maps: In theory, a costmap designed for ground vehicles could map the surface of the ground in 3D. In practice, however, in order to simplify the operations on the costmap, it is common to project it on a plane that approximates the ground surface, called ground plane. This is a local approximation, and different costmaps should be used to represent large areas with varying inclination.

In order to find the ground plane, we need to first define the ground. A general solution to this problem is not straightforward, so in this document we propose a solution for areas that are roughly horizontal, which is well-suited for our need. Our solution is to start by filtering out the points in the 3D map based on height in the East North Up (ENU) reference frame. This process filters out all of the tall objects

[1] https://github.com/CranfieldUniversitySignalsAndAutonomy/costmap_generation.git

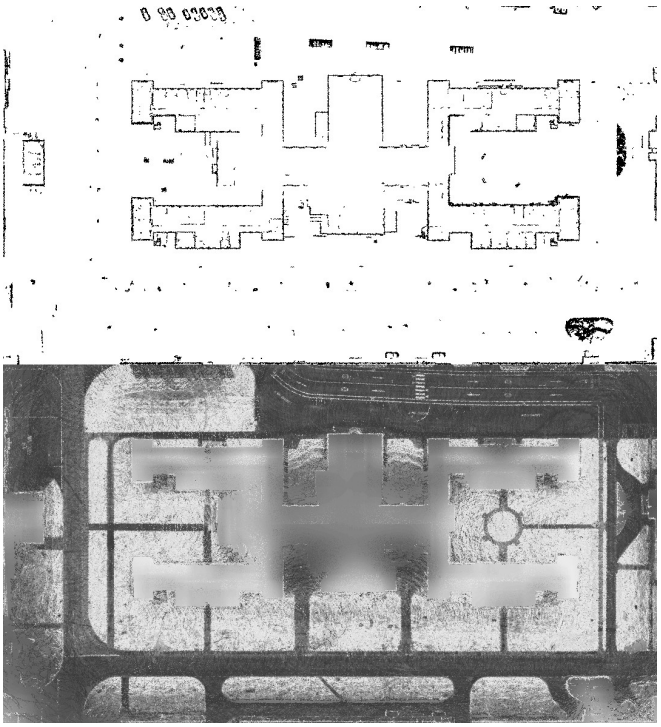

Fig. 3. The traversability map and the intensity map of the demonstration area, the value of unknown cells has been interpolated from neighbouring cells for the intensity map (for visibility reasons), while it has been assigned as traversable in the traversability map

in the scene, leaving just the ground and the lower parts of the objects. Now it is possible to calculate the plane that best fits the remaining points, and we consider that to be the ground plane for the area.

Once the ground plane is defined, it is possible to identify traversable and non-traversable areas based on the height gradient. Thanks to our software, we were able to automatically generate three types of 2D maps:

- a traversability map, which is binary and defines traversable and non-traversable areas
- a height map of the ground
- an intensity map of the ground, where by intensity we mean the intensity of the reflection of the lidar beam

The height map that we generated has very faint features due to the noise in the measurements and the smoothing introduced by the mapping process. Because of this, despite trying different edge-detection algorithms, we were unable to reliably detect the kerb at the side of the road. The traversability map and the intensity map are shown in Figure 3. The traversability map represents unknown areas, such as occluded walls or the interior of buildings, as traversable. This is because there are a lot of cells in the 2D map that have an unknown status (i.e. they have not been scanned by a lidar beam) and it would be too restrictive to assume that there is an obstacle on every unknown cell, so we decided to assign traversability by default and to rely on manual marking of the obstacles that were not recorded. In our case, this was only a matter of drawing the occluded walls.

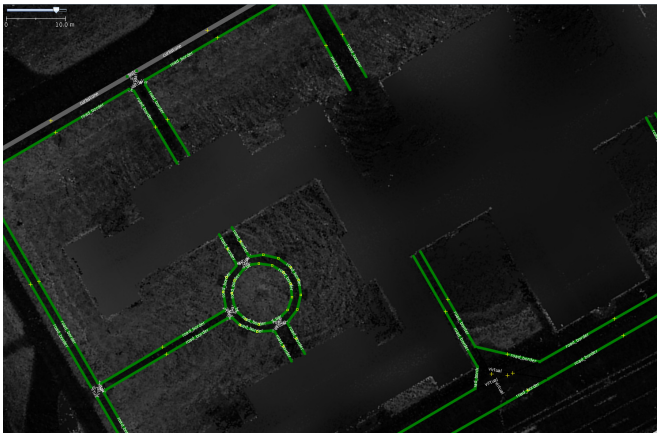

Fig. 4. The vector map created on top of the intensity map, as seen in the map editor JOSM

2) Vector Map: In order to represent and to classify the different traversable areas in the map, we looked at some of the public map formats available [22], [23]. We decided to use *lanelet2* [23], because it is a light map format that provides a representation of a wide variety of features of structured environments, and it also offers the possibility to define unstructured areas. Additionally, custom attributes can be used to specify any characteristic of any area or feature in the scene. Using the editor JOSM and the lanelet2 library, we were able to mark the pavement and the kerb on top of the intensity map, as shown in Figure 4.

3) Static Layer: Starting from the vector map and the traversability map, the static layer of the costmap can be generated through the following steps:

- definition of the cost to add to each cell if this lays within a certain type of lanelet or if it contains a certain type of feature in the vector map
- definition of the cost of hitting a non-traversable object; this is usually a hard constrain (lethal cost) for the path planner
- determination of the associations between the cells in the traversability map, the lanelets and the features in the vector map
- accumulations of the costs associated to each cell

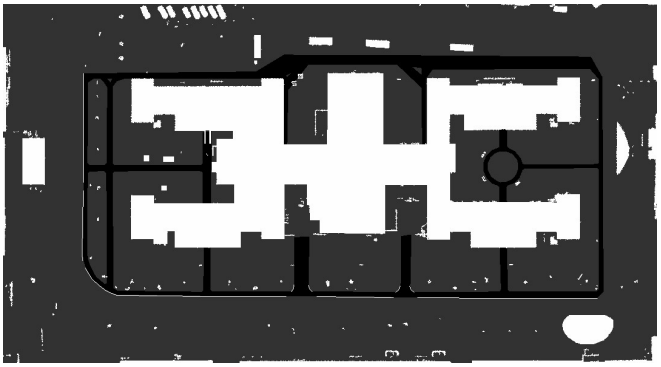

Fig. 5. The static layer of the costmap

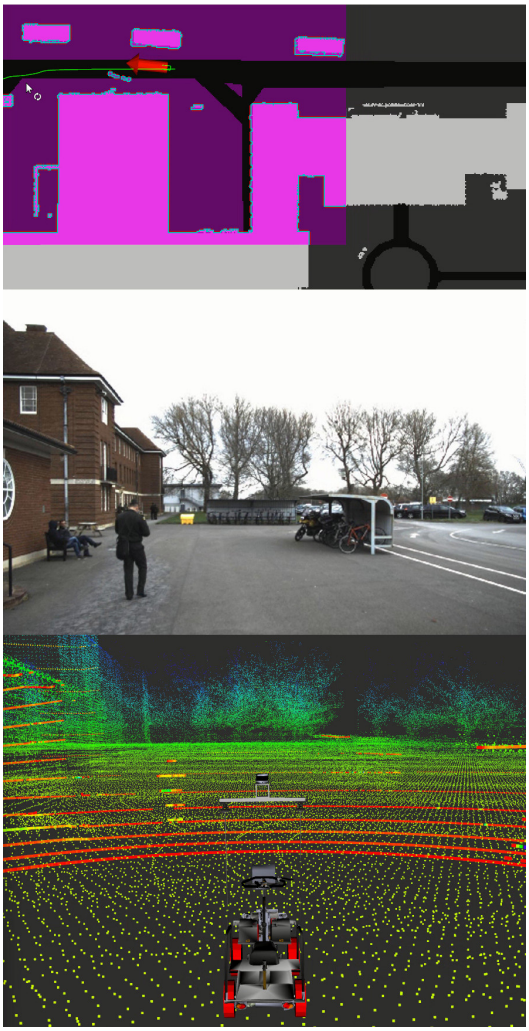

Fig. 6. A snapshot of the user interface; on top there is a 2D view of the global costmap, represented in grayscale, the local costmap, represented with different shades of pink, the target path, in green, and the vehicle pose, represented by the red arrow; at the centre there is the camera view of the same scene; at the bottom there is a reconstructed 3D view with the lidar scan, represented by thick points/lines, the 3D map, represented by finer points, and a 3D mesh of the vehicle, in the foreground

Our Matlab library automates the last two steps of the process. The result is shown in Figure 5.

4) Path Planning: There are several path planning libraries that are compatible with ROS [24], [25], but only a few of them are specifically designed for non-holonomic ground robots, and even fewer of them are regularly maintained. We decided to use the ROS *navigation* package [24] because of several interesting features:

- implementation of a layered costmap with different types of layers
  - *static map layer*, to represent static objects in the scene
  - *obstacle map layer*, to represent dynamic objects as detected by the sensors; this layer can be generated from a 3D occupancy grid that is populated directly from the output of the lidar sensor
  - *inflation layer*, to represent the preference of maintaining some distance from the obstacles
- implementation of costmaps that can move with the robot
- two layers of planning, global planner and local planner, operating on two separate costmaps

Configuring the navigation package was straightforward and we were quickly able to test the operations of the vehicle on the ground.

Figure 6 shows a snapshot of the dataset recorded during the early phase of testing. The top part of the figure shows the operations of the path planner. The red arrow indicates the vehicle pose, while the vehicle footprint is represented by a tiny green rectangle at the base of the arrow. The costmap in the background, in greyscale, is the input of the global planner and it is used to calculate the global path from the vehicle pose to the goal pose, which is shown as a green line. The global costmap represents the whole demonstration area and does not move in time. On top of it, the pink colouring represents the cost of the local costmap, used as the input of the local planner to avoid dynamic obstacles in the vicinity of the vehicle. This costmap is a square of $60\,\mathrm{m} \times 60\,\mathrm{m}$ that moves with the vehicle. In front of the vehicle, it is possible to see a series of circles representing a person walking ahead. The trail behind the person is caused by the lack of lidar beams passing through the previously-occupied cells, which prevents their clearing though raytracing. This is a problem of using low-resolution lidar. However, thanks to the continuous movement of the vehicle, the cells are usually cleared in a couple of seconds.

IV. CONCLUSIONS AND FUTURE WORK

We presented a complete framework for autonomous navigation on pedestrian areas, based entirely on open-source software. The framework has been used for the development of a prototype mobility scooter, whose system and software architecture have been described in detail.

The initial experiments provide already some key directions for future work:

- the resolution of the lidar is not sufficient to reliably detect occupancy, so we are planning to use the output of the cameras to improve the detail of the point cloud [9], [10]
- the blind spot underneath the lidar sensor needs to be covered by additional sensors
- the path planner does not consider the kinematics of the vehicle, which could be a problem in tight environments; we should integrate a more suitable planning algorithm, like hybrid A* search [19], into the same framework

We are at the beginning of the testing phase and we expect over the next few months to produce data for a quantitative analysis of the performance of the proposed architecture.

ACKNOWLEDGMENT

The authors would like to thank TMSUK Company Limited for supporting this project.

REFERENCES

[1] Research Institute for Consumer Affairs (Rica), "Mobility scooters: a market study," Research Institute for Consumer Affairs (Rica), Tech. Rep., 2014.

[2] United Nations, *World Population Ageing 2015*. 2017, p. 159.

[3] A. Bingham, X. Hadoux, and D. K. Kumar, "Implementation of a safety system using ir and ultrasonic devices for mobility scooter obstacle collision avoidance," *ISSNIP Biosignals and Biorobotics Conference, BRC*, pp. 1–5, 2014.

[4] D Eck, T Heim, R Hess, and K Schilling, "A Narrow Passage Assistance Functions on a Mobility Scooter for Elderly People," in *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 383–388.

[5] Y. Morales, E. Takeuchi, A. Carballo, W. Tokunaga, H. Kuniyoshi, A. Aburadani, A. Hirosawa, Y. Nagasaka, Y. Suzuki, and T. Tsubouchi, "1Km autonomous robot navigation on outdoor pedestrian paths "Running the Tsukuba Challenge 2007"," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 219–225, 2008.

[6] H. Date and Y. Takita, "Real world experiments of an autonomous mobile robot in the pedestrian environment," *ICARA 2011 - Proceedings of the 5th International Conference on Automation, Robotics and Applications*, pp. 413–418, 2011.

[7] M. Hirai, T. Tomizawa, S. Muramatsu, M. Sato, S. Kudoh, and T. Suehiro, "Development of an intelligent mobility scooter," *2012 IEEE International Conference on Mechatronics and Automation, ICMA 2012*, pp. 45–52, 2012.

[8] *TMSUK Rodem*. [Online]. Available: `http://www.tmsuk.co.jp/en/rodem/` (visited on 04/09/2019).

[9] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-November, pp. 2181–2188, 2016.

[10] H. Courtois and N. Aouf, "Fusion of stereo and Lidar data for dense depth map computation," *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems, RED-UAS 2017*, pp. 186–191, 2017.

[11] M. Quigley, K. Conley, B. P Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y Ng, *ROS: an open-source Robot Operating System*. 2009, vol. 3.

[12] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[13] *velodyne ROS package*. [Online]. Available: `http://wiki.ros.org/velodyne` (visited on 04/10/2019).

[14] *xsens_driver ROS package*. [Online]. Available: `http://wiki.ros.org/xsens_driver` (visited on 04/10/2019).

[15] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[16] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[17] T. Stoyanov, M. Magnusson, and A. J. Lilienthal, "Point set registration through minimization of the L2distance between 3D-NDT models," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5196–5201, 2012.

[18] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, 2013.

[19] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[20] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," *IEEE International Conference on Intelligent Robots and Systems*, no. Iros, pp. 709–715, 2014.

[21] V. Smith, J. Malik, and D. Culler, "Classification of sidewalks in street view images," *2013 International Green Computing Conference Proceedings, IGCC 2013*, pp. 1–6, 2013.

[22] M. Haklay and P. Weber, "OpenStreet map: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[23] F. Poggenhans, J. H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1672–1679.

[24] D. V. Lu and W. D. Smart, "Towards more efficient navigation for robots and humans," *IEEE International Conference on Intelligent Robots and Systems*, no. c, pp. 1707–1713, 2013.

[25] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!" *IEEE Robotics and Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

https://dspace.org/

Autonomous navigation for mobility scooters: a complete framework based on open-source software

Cecotti, Marco; Kanchwala, Husain; Aouf, Nabil

2019-11-28
IEEE
https://doi.org/10.1109/ITSC.2019.8917469
Cecotti M, Kanchwala H, Aouf N. (2019) Autonomous navigation for mobility scooters: a complete framework based on open-source software. In: 2019 IEEE Intelligent Transportation Systems Conference - ITSC 2019, Auckland, 27-30 October 2019

Downloaded from a DSpace repository. https://dspace.org/