2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

An Autonomous Vehicle for Parcel Delivery in Urban Areas

Alexander Buchegger[1,2], Konstantin Lassnig[1], Stefan Loigge[1,2], Clemens Mühlbacher[1], and Gerald Steinbauer[2]

Abstract— **The flexible and individualized transportation of goods is a central task of today's e-economy. In urban and highly populated areas autonomous electric vehicles are a promising solution for this task while simultaneously addressing ecological issues. While in indoor environments transport robots are well adopted, autonomous transport vehicles are hardly seen outdoors.**

In this paper, we aim at this gap and adapt and transfer concepts usually used in robotics to autonomous vehicles for an outdoor environment. We present an autonomous vehicle that is able to safely navigate in urban environments while able to deliver parcels efficiently. In particular, we will discuss a scalable and robust mapping and navigation process that forms the basis for the capabilities of the delivery vehicle. Moreover, we show preliminary results of a deployment of the system in two urban scenarios.

I. Introduction

The flexible and cheap transportation of goods is a central part of today's e-commerce. Goods which are sold over the Internet heavily depend on transportation costs in particular on the last mile. Moreover, the increasing amount of goods that needs to be handled has to be addressed. For instance the Austrian postal service expects an annual increase of 15 % in the number of parcel. Today's supply chain requires a very dense distribution network and relies on the fact that sending a lot of packages on the same route is cheap. The larger the number of goods for one route, the cheaper it becomes. This effect contrasts with the increasing need for transporting goods to a single customer in a flexible and individualized way. Such a transportation is characterized by a small number of goods for one transportation route. To address this, autonomous intelligent transport vehicles offer a possible solution in urban and more densely populated areas. Using such systems, the transportation can be performed in a flexible way. Additionally, by using an electric vehicle, the ecological footprint such as pollution and noise can be reduced in city centers.

The use of a robot fleet for transportation tasks is standard for warehouse environments nowadays [1], [2], [3]. These robot systems allow transporting goods in a warehouse in a flexible way without the need of adapting the warehouse. This is achieved by using algorithms for localization and navigation in known structured environments [4]. These algorithms are hardly seen for large outdoor environments such as a city center and its surroundings.

To allow autonomous transport vehicles to be used for transportation tasks in large-scale outdoor environments proven approaches from the robotics domain needs to be applied and transferred to these new environments. In this paper, we present an integrated autonomous transport vehicle which addresses these problems and is able to deliver parcels in urban environments such as city centers automatically. The developed transport vehicle is based on a commercial electrical personal vehicle. It was adapted for autonomous control and equipped with improved navigation skills for outdoor environments based on a topological navigation approach. The integrated vehicle was evaluated in realistic delivery use cases where parcels are delivered autonomously to addresses in a larger urban area.

The main contributions of this paper are: (1) the adaptation of well-known algorithms for robot navigation for large-scale urban environments, (2) an integration of these algorithms in a commercially available electrical vehicle, (3) the improvement of the robustness of the approach by integrating additional sensors like GPS and pre-knowledge from OpenStreetMap (OSM), and (4) an evaluation of the autonomous delivery concept in real urban environments such as an university campus or a city center.

The remainder of this paper is organized as follows. In the next section we discuss some related research. In section III we introduce the adapted transport vehicle. In the next section, we will discuss the software system used to perform transportation tasks in an outdoor environment. The proceeding section presents results on how the vehicle performs autonomous navigation in a large-scale environment. Finally, we conclude the paper and point out some future work.

II. Related Research

We start our discussion of related research with the robot presented in [5]. The robot could take a long tour through Munich without a prior created map or GPS information. The robot was using its sensors to navigate locally in a safe manner and asked humans for information about the global direction. This was done by approaching humans and the recognition of basic commands to derive the direction of the desired destination. In contrast, our vehicle has a prior created map which allows it to move autonomously without asking for directions. This is also desirable in the case of a transport robot which should transport goods to a customer.

In [6] the method to deal with large maps was presented. The authors used a topological map to allow efficient representation of large areas. The vertices in the topological map

[1] Alexander Buchegger, Konstantin Lassnig, Stefan Loigge, and Clemens Mühlbacher are with Arti Robots, Graz, Austria {a.buchegger, k.lassnig, s.loigge, c.muehlbacher}@arti-robots.com

[2] Alexander Buchegger, Stefan Loigge, and Gerald Steinbauer are with the Institute of Software Technology, Graz University of Technology, Graz, Austria steinbauer@ist.tugraz.at

978-1-7281-0323-5/18/$31.00 ©2018 IEEE

are spots of interests such as a square or a crossing. The edges represent paths between these places. For each edge, a traversal behavior is defined. Thus, one can use different behaviors to perform the traversal. With the help of this method, the robot could drive autonomously in a park. In contrast, our vehicle uses a topological map which contains enough information to allow the robot to be always localized also besides very prominent places. Furthermore, our robot uses a denser road map allowing it to plan its route more accurately.

Very closely related work to ours was presented in [7]. The robot navigated more than 3km in the city of Freiburg in an autonomous fashion. To localize itself, the robot used a topological map where each vertex in the graph contains a map of one part of the environment. In contrast, our approach additionally used the GPS signal for estimating the robot pose within the particle filter. To navigate the robot, the method presented in [7] created a high-level plan using the graph of the topological map. Each vertex is connected to those vertices in the graph which allow moving between these two locations. Thus, using this graph, the robot can derive a simple high-level plan for the navigation. The robot uses a planner on grid map basis to navigate between different vertices of the topological map. This contrasts with our approach as we use a fine-grained road map for the high-level planning which allows us to choose the path more precisely.

III. The Electrical Vehicle

Fig. 1. Jeflyer - commercial electrical vehicle (left). Vehicle after adaptation for autonomous navigation.

The base of the transport vehicle is a street-legal, electric vehicle called Jetflyer by i-Tec Styria. The vehicle is already used for manual parcel delivery. Figure 1 depicts the original and adapted vehicle. The vehicle was adapted for drive-by-wire, was equipped with sensors and computational resources, and holds storage compartments for the individual parcels that can be opened with a code. The code is sent to the recipient via SMS to allow to retrieve the parcel.

To use this vehicle for autonomous driving, it was necessary to adapt the steering, the braking system, and the motor driver control to enable drive-by-wire. An additional control unit serves as the central connection between the vehicle and the main computer. It receives driving commands from the computer and controls throttle, brake, and steering. Moreover, it collects required state information from the vehicle like its current velocity, steering angle, and battery status, and sends this information back to the computer.

Besides the adaptation of the vehicle, different sensors were mounted. They include three laser scanners for localization and obstacle detection. Two laser scanners (Sick LMS291) are mounted horizontally in the front (front laser) and back (back laser) to detect static and dynamic obstacles like poles, cars, bicycles, and pedestrians. Please note that the scanner is laser class 1 (eye-safe). The third laser scanner (top laser) of type Sick LMS100 is mounted horizontally on a mast at the height of 1.9 meters. This laser scanner is used to localize the vehicle in relation to a given map. Besides the laser scanners, the robot has a Global Positioning System (GPS) receiver to support localization and mapping. To improve the accuracy of the vehicle's odometry, an Inertial Measurement Unit (IMU) is mounted inside the vehicle.

IV. System Overview

To perform transportation tasks, the robot uses the system architecture depicted in Figure 2.

The user (e.g., postman) interacts with the system via a web-based Graphical User Interface (GUI), depicted in Figure 3. Via this GUI, the user can control the operating mode of the robot (pause, deliver, or return home) and enter the delivery addresses for the parcels in the vehicle's storage compartments. The GUI interacts with the high-level state machine, which controls the behavior of the vehicle, and with the mission planner, which translates the delivery addresses into geographic coordinates using OSM data and determines an efficient order in which the parcels will be delivered. The GUI in return displays the current operating mode and any issues relevant for the user. The user can furthermore check the status of the deliveries - whether a delivery is still in the queue, is currently being processed, was completed successfully, or was canceled due to an error. The GUI has been realized using a slim web-server implemented in Python and content generated using Javascript.

The high-level state machine, when in delivery mode, fetches the next delivery mission from the mission planner, and commands the topological planner to plan and execute a traversal to the target location. Currently delivery missions are selected based on first-come first-serve. Optimization of delivery routes is subject to future work.

The topological planner creates a high-level path using the topological roadmap, and in turn commands the lower-level navigation system to traverse this path. The navigation stack is explained in more detail in the next section.

V. Navigation

In the previous section, we have discussed how the control system is composed and how the user interacts with the system. The robot is designed to deliver goods to different locations and therefore requires a reliable and efficient navigation system. This is achieved by first creating a map of the environment which is afterward used to localize the robot. The acquired location of the robot is then used to

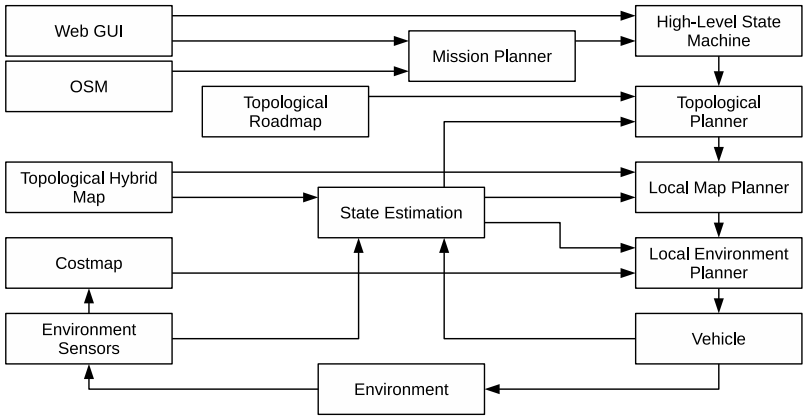

Fig. 2. Overview of the transport vehicle's control system.

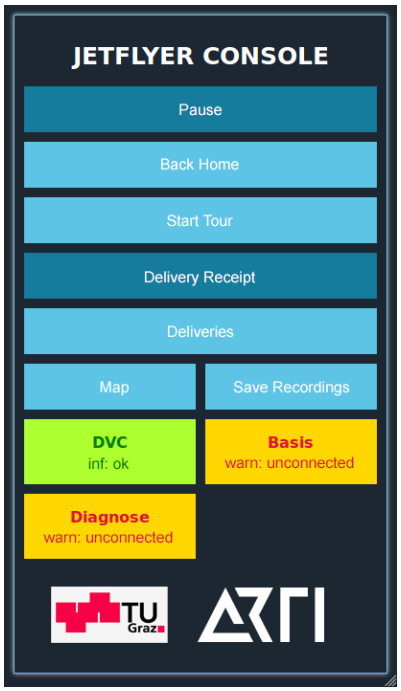

Fig. 3. The vehicle's web-based GUI.

plan a path to the required destination. The planned path is continuously executed and corrected due to observations from the environment, which is necessary to avoid obstacles reliably.

A. Mapping

The creation of a map which is later used for localization consists of two stages. The first stage creates a pose graph [4] containing all positions, the robot has traveled during map creation together with the acquired sensing information. The second stage uses the pose graph to generate a set of sub-maps, where each sub-map is a gridmap which covers 50 $\times$ 50 meters of the environment. Each of these sub-maps is part of the topological map [7] which defines how the sub-maps are related to each other. The advantage of using sub-maps is that the robot only needs to cope with a small map to estimate its current location but is still able to determine its absolute location with respect to the world. This approach allows for a robust and efficient application to large-scale environments.

The pose graph consists of vertices representing poses of the robot and edges defining relative movement between these poses or the relation to a global position reference. The creation of vertices is triggered every time the robot has traveled at least $1\ m$ or has rotated at least $0.5\ rad$. When a new vertex is created, the relative displacement between this new vertex and the previously added vertex is calculated via a laser scan matching [4] algorithm. The new vertex is added to the pose graph together with an edge which defines the relative displacement calculated and an estimation of the error of that displacement.

Each time a new vertex is added to the pose graph, the nearby surroundings are checked whether two vertices are not directly connected (or connected by a predefined number of hops) but close enough (in the Euclidean distance) for potential loop closures. For such pairs, a loop closure check is performed. This test tries to establish a connection between the two poses by using a more accurate but also more expensive laser scan matching search to determine a potential relative displacement. If the test succeeds, a new edge describing this relative displacement is added to the graph. This allows the mapping process to recognize places which were visited before and allowed to create a globally consistent map even for large environments. If a new edge has been added, the loop closure check is performed again to determine whether another pair of vertices is now close enough to be connected. Thus, one loop closure often causes a cascade of edges to be added, which improves the overall quality of the map.

As the relative displacement of two vertices is calculated through laser scan matching, it is important that this matching is fast and accurate. To address both requirements, we use a cascade of three different scan matchers. If the first one fails, the second one is used, and so on. This allows a fast scan matching result, but also gives a high chance that

a correct match was found.

The first scan matcher uses a k-d tree [8] to determine the score of a potential transformation. For each laser scan point, it looks for the closest laser scan point of the previous measurements. The average distance of the closest points determines the score of a transformation. To determine the best transformation, a greedy hill climbing algorithm is used.

The second scan matcher uses a matching routine further described in [9]. The algorithm uses points of the previous scan which are matched to lines of the current scan to derive an equation system. This equation system can be solved numerically to determine the best transformation.

The third scan matcher uses the technique described in [10]. This scan matcher uses a grid to save the previous laser scan measurements. Using this grid, a score can be computed for a transformation by determining the value in the grid cell corresponding to the position of a laser point. Afterwards, a greedy hill climbing algorithm is applied to determine the transformation with the highest score.

Besides the relative displacement between poses, which is determined using the laser scan matcher, we use two distinctive methods to determine a global reference. This global reference is used to further constrain the poses in the pose graph.

The first global reference is the GPS position which is received by a sensor mounted on top of the robot. The GPS fix associated with the first vertex in the pose graph is used as a local reference point. Every further GPS fix is related to this local geo-reference. This allows to directly relate the Cartesian position of the poses within the pose graph to the GPS fixes. These relations are represented as additional constrains in the graph.

The second global reference is generated with the help of OSM [11] data. For each pose, the laser scan data are matched against the OSM data. As a matching seed, the geo-referenced position of the pose graph poses is used. Afterwards, greedy hill climbing is applied to determine the global pose. To ensure a proper seed position, the OSM matching is only applied after the robot has traveled at least 20 m. This ensures that enough GPS positions were received and the pose graph is sufficiently geo-referenced. This approach allows to reuse publicly available map data and to refine them with actual measurements from the environment.

After the integration of new nodes and vertices, the resulting graph is optimized subject to the displacement error of vertices using the g2o optimization framework [12] to obtain a globally consistent graph.

Once the pose graph is finished, the corresponding sub-gridmaps are created using the simple mapping with known poses approach [4]. For each sub-map, all relevant vertices are collected (i.e., poses where laser scans were collected) and the related laser scans are included into the local sub-gridmap. A new sub-gridmap is added if the related vertex is far enough from neighboring vertices but allow still a sufficient overlap of the sub-gridmaps. The latter is important to allow an efficient switching of the maps during navigation.

To allow the robot to manage the sub-gridmaps properly, each sub-gridmap is part of the topological map graph. Each vertex of this graph is an occupancy grid map, and each edge defines a possible traversal from one sub-map to another sub-map. The traversability is checked by combining both maps and planning a path from the center pose of one map to the center pose of the other map. Figure 4 shows a part of such a graph. Please note that grid 11 is connected neither to grid 3 nor to grid 4, as it is not possible to move between grid 11 and grid 3 or grid 4 due to the wall.

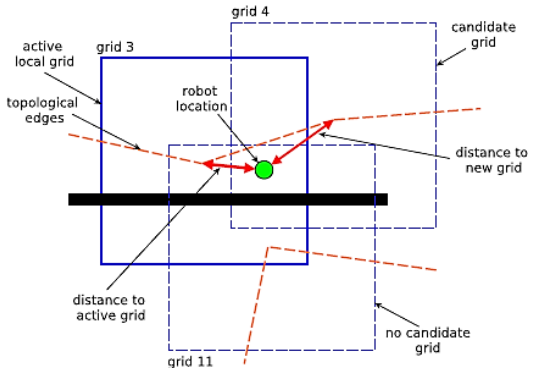

Fig. 4. Grid selection for the localization, together with the topological map [13].

B. Localization

With the help of the topological map, the robot can localize itself. The initial position of the robot is set by the user. Starting with this initial estimate, the robot uses a particle filter approach [14] in order to continuously localize itself on the sub-gridmap respectively on the global map as the relations between the sub-maps are known.

To be more precise, we use an enhanced version of Adaptive Monte Carlo Localization (AMCL) [4]. We improved the robustness of AMCL by considering GPS signals as well as using the command velocities sent to the robot to trigger particle updates.

The GPS signal was incorporated in the same way as it was for the pose graph. The first GPS position received in the mapping process was used as an anchor to transform all following GPS positions into a local geo-referenced coordinate system. This information is used to adjust the particle weights in the filter for each received GPS position according to the Cartesian displacement between the GPS signal and the particle's pose. As the GPS positions showed a large variance to the real pose, we assumed a normal distribution with covariance of 10 m to ensure proper weightage, and to avoid possible divergences from the true position. Thus, the GPS positions were used as a hint to recover from localization errors.

Due to a shortcoming of the vehicle's motor controller, the robot did not receive odometry information for very slow movements. These movements sometimes occurred in situations where the robot re-adjusted its path to avoid a dynamic obstacle. To force a particle update in these

situations, we used the velocity commanded to the vehicle to propagate the particles and afterward performed the particle weightage as usual. This allowed the robot to be precisely localized even with missing odometry information during slow movements.

As the particle filter mainly performs its weightage based on the map, it is important for the localization that the robot uses the best available sub-map. Therefore, the robot performs a check after every pose update to find the best sub-map. This is done by considering all sub-maps which are adjacent to the current one in the topological map graph. The best one is chosen by comparing all distances between the current robot pose and the center of every adjacent sub-map. To avoid frequent map changes, the current map is only changed if the distance to an adjacent sub-map's center is at least $90\ \%$ smaller than the distance to the current sub-map's center. The advantage of this approach is that most of time the localization runs on a single sub-map.

C. Path Planning and Execution

With the help of the particle filter, the robot can determine its current location. Using this information, the robot is capable of planning paths. To determine where to drive to, the robot uses the postal address which was entered by the user in the web interface. The postal address is looked up in the OSM data to find the corresponding GPS coordinates. If no GPS coordinates are specified in OSM directly, the contour of the building is used instead. The center of gravity of the contour is calculated and used as coordinates. After the GPS coordinates are determined, the coordinates are transformed to the local, geo-referenced system as it is done for the localization.

To move from the current location to the specified goal location, we follow a three-level planning approach. The first level is the topological planner. This planner uses a road map to calculate a route from the target to the goal location. The road map is created along the path the robot has traveled during map creation. This was a decision made to ensure that the robot only drives on known and safe routes, avoiding non-traversable or dangerous areas (i.e., these areas had been avoided during mapping). To create a topological path, the current robot position is connected to the closest road map location, and additionally, the goal location is connected to its closest road map location. To finally generate the topological path, the A^* search algorithm is applied on the road map.

A further safety mechanism was introduced to prevent the robot from making detours too far away from the topological plan. A corridor of $2m$ around the topological path is generated, forcing the robot to stay close to the topological path. This concept further avoids producing plans in the second and third stage too far away from known safe paths. This helps the robot to avoid non-traversable and dangerous areas (e.g., sidewalks and street-car tracks). Figure 5 shows a road map, the topological plan, the local plan and the corresponding corridor.

After a topological path is found, the robot gradually plans from its current location to the next point in the topological path. If the robot is near a point (closer than $1\ m$), the point is removed from the topological plan, and the robot targets the next point. To plan from the current position to the active goal point, a global plan is created with the help of the Search-Based Planning Library (SBPL) planner [15], considering the sensor readings accumulated over time, to prevent collision with objects. The planner uses precomputed motion primitives to find a valid path. This restricts the planner only to produce motions which can be executed by the actual vehicle kinematics (Ackerman steering).

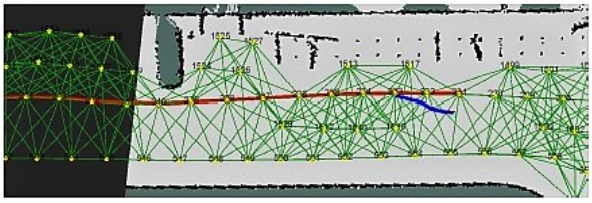

Fig. 5. A road map (green), the topological plan (red), the local plan (blue) and the corresponding corridor [13].

While following this path, the robot avoids local obstacles at the same time. This is achieved by using the Dynamic Window Approach (DWA) planner [16]. This planner samples possible trajectories and computes their scores based on the distance to obstacles, the desired path, and the expected progress. To address the Ackerman steering, all samples which do not fulfill a minimum turn radius criterion are removed from the sample sets.

VI. Results

In the previous section, we have discussed how the robot executes an autonomous parcel delivery. To demonstrate these capabilities, the robot performed several delivery tasks in two different urban areas. Test area 1 is depicted in Figure 6 and represents a part of the university campus. This area was used for the initial test because it is closed for public traffic.

Fig. 6. Test area 1: university campus ((c) Google Maps).

Test area 2 is depicted in Figure 7 and represents the city center of Graz. This area was used for the final test because it is a pedestrian zone, but there are numerous pedestrians

and cyclists around as well as there is traffic by delivery trucks and street-cars. This choice was made because of two reasons. First the areas cover nicely the intended use case of urban parcel delivery. Second the legal situation in Austria autonomous driving on public roads is restricted to only a few cases such as driver assistance systems on highways.

Fig. 7. Test area 2: city center Graz ((c) Google Maps). The red C marks the starting courtyard.

In a first experiment, we evaluated our proposed mapping approach for the two test areas. We drove the robot manually on usual delivery paths that allows the vehicle to reach a decent number of delivery addresses. Figure 8 shows the resulting global pose graph and a global gridmap for test area 1. Please note that this map was only used for inspection and not for navigation. Figure 9 shows the resulting global pose graph and a global gridmap for test area 2. The maps are consistent and allow decent navigation of the vehicle.

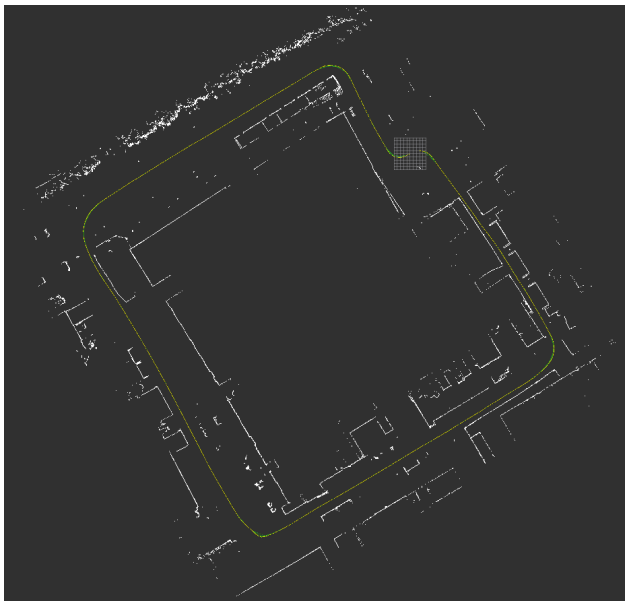

Fig. 8. Map of university campus. Global gridmap (white). Pose graph with odometry edges (Green). GPS vertices (blue).

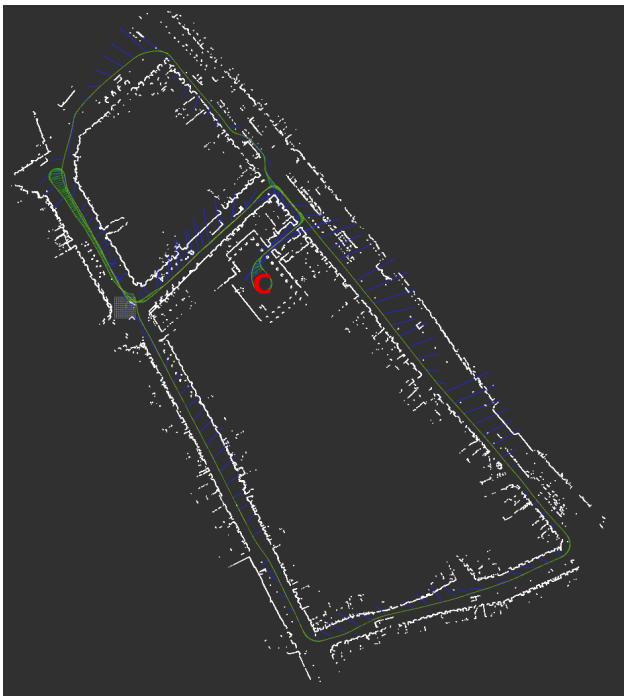

Fig. 9. Map of city center Graz. Global gridmap (white). Pose graph with odometry edges (Green). GPS vertices (blue). The red C marks the starting courtyard.

Figure 10 shows the resulting topological roadmap and sub-map for test area 2. The mapping step is performed while driving the robot manually to safe and interesting places. It forms the base for the automated navigation.

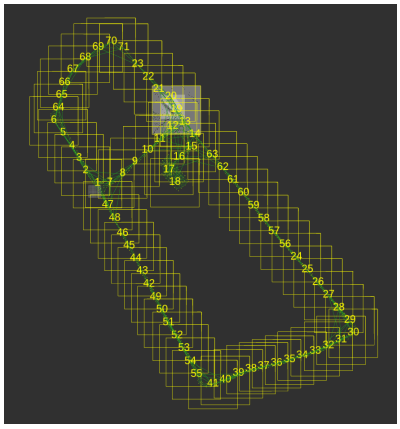

Fig. 10. Topological map of city center Graz. Global roadmap (green). Local gridmaps (yellow).

In the second set of experiments, we evaluated if the vehicle is able to perform delivery tasks comprising of four individual postal addresses autonomously. We use the evaluation criterion if the vehicle autonomously drove to all given addresses and correctly reported the status of the delivery.

We conducted three different delivery experiments. The first delivery route covers a driving distance of $450\ m$. On this route, the robot was commanded to four addresses on

the university campus. The robot drove this route 40 times, encountering one problem in one trip. The problem was a special turn in front of a building (at the lower left corner). The robot had to turn-in-place to move on from one building to another. This problem was mainly caused by the restricted driving corridor, which prevented the robot from executing complex maneuvering tasks where more safe space is needed.

The second route had a length of $1000\ m$ and was executed seven times. The route included six different delivery locations and an extended map. During this experiment, the vehicle became de-localized two times where it was stopped manually. This failure was encountered in an open area and was caused by a combination of bad odometry and lack of features for matching laser scans in this area at the same time. Unfortunately, the GPS guesses were not accurate enough to allow automated re-localization.

The third route was conducted in the city center and comprised four delivery addresses. Please note that the vehicle always started in the courtyard in the center of the map. This route had a total length of $960\ m$ and was executed nine times. Although, the environment was crowded during these experiments, no problems occurred. Even though the GPS signal was very bad (more than $10\ m$ discrepancy between actual and reported position), the vehicle was always localized correctly due to the high number of features for laser scan matching in the city center.

To conclude the mapping and navigation approach worked very well. Neither student gatherings at the university campus, nor rush-hour traffic in the city center caused the vehicle to fail in the fulfillment of the tasks. Additionally, the vehicle was well-accepted by other road users. Also, the public feedback about its capabilities was very positive.

VII. Conclusion and Future work

The transportation of goods is an essential part of today's e-economy. The transportation often takes place in outdoor environments by delivering goods to customers. To provide cost-efficient and flexible deliveries for the last mile and address the ecological issue, medium-sized electric vehicles are a promising solution. While transport robots are well adopted in indoor environments, they are hardly seen in outdoor environments. The work presented in this paper aims to close that gap.

In this paper, we presented an autonomous transport vehicle which is capable of navigating in large-scale urban environments. The vehicle is based on a commercially available electric vehicle, which had been adapted for autonomous operation. We developed a topological mapping approach comprising a global roadmap and local gridmaps which can map large-scale outdoor environments. Moreover, we developed a hierarchical navigation approach that allows the vehicle to safely navigate in urban areas while supporting long routes as well as respecting the special kinematics of the vehicle. Addressing a parcel delivery use case, an infrastructure was added that allows a user to command delivery routes to the vehicle. The vehicle was evaluated in 2 urban settings. The evaluation showed that the developed vehicle could reliably and safely navigate in crowded areas and that the concept is well recognized by the professional user and bystanders.

In future work, it is necessary to add more sensors and algorithms to obtain more information about the context such as recognizing the behavior of others or special locations such as crossings. Although the vehicle could navigate robustly, there are situations where the robot lost a lot of time being blocked in a narrow street with a truck approaching instead of actively resolving the situation (e.g., let the truck out first). Moreover, the scalability of the approach in terms of size of area and number of robots needs to be investigated.

Acknowledgment

This work was carried out in cooperation with the Österreichische Post AG, the Energie Steiermark, Institute of Automotive Engineering of Graz University of Technology and i-Tec Styria GmbH. In memoriam of Gernot Hiebler.

References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *19th national conference on Innovative applications of artificial intelligence - Volume 2*, ser. IAAI'07. AAAI Press, 2007, pp. 1752–1759.

[2] E. Guizzo, "Three Engineers, Hundreds of Robots, One Warehouse," *Spectrum, IEEE*, vol. 45, no. 7, pp. 26–34, 2008.

[3] C. Mühlbacher, S. Gspandl, M. Reip, and G. Steinbauer, "Improving dependability of industrial transport robots using model-based techniques," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3133–3140.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[5] A. Bauer, K. Klasing, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss, "The autonomous city explorer: Towards natural human-robot interaction in urban environments," *International Journal of Social Robotics*, vol. 1, no. 2, pp. 127–140, 2009.

[6] K. Košnar, T. Krajník, V. Vonásek, and L. Preucil, "Lama-large maps framework," in *Proceedings of Workshop on Field Robotics, Civilian-European Robot Trial*, 2009, pp. 9–16.

[7] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "Autonomous robot navigation in highly populated pedestrian zones," *Journal of Field Robotics*, vol. 32, no. 4, pp. 565–589, 2015.

[8] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sept. 1975. [Online]. Available: http://doi.acm.org/10.1145/361002.361007

[9] A. Censi, "An ICP variant using a point-to-line metric," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[10] K. Ryu, L. Dantanarayana, T. Furukawa, and G. Dissanayake, "Grid-based scan-to-map matching for accurate 2d map building," *Advanced Robotics*, vol. 30, no. 7, pp. 431–448, 2016.

[11] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org , 2017.

[12] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011.

[13] K. Lassnig, "An autonomous robot for campus-wide transport tasks," Master's thesis, Faculty of Computer Science and Biomedical Engineering, Graz University of Technology, 2016.

[14] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[15] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev, "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014.

[16] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.